

**КИЇВСЬКИЙ СТОЛИЧНИЙ УНІВЕРСИТЕТ
ІМЕНІ БОРИСА ГРІНЧЕНКА**

Костюк Ю.В., Складанний П.М.

**ЗАХИСТ ІНФОРМАЦІЇ
В КОМП'ЮТЕРНИХ СИСТЕМАХ
ТА МЕРЕЖАХ**

Підручник

*Частина I. Теоретичні та методологічні
основи захисту інформації в комп'ютерних
системах та мережах*

Київ 2026

УДК 004.056:004.7(075.8)

*Рекомендовано Вченою радою Факультету інформаційних технологій та математики
(протокол № 1 від 14 січня 2026 року)*

Автори:

Костюк Ю.В., Складанний П.М.

Рецензенти: С. В. Толюпа, д-р техн. наук, професор, професор кафедри кібербезпеки та захисту інформації, Київського національного університету імені Тараса Шевченка;
І. Р. Опірський, д-р техн. наук, професор, завідувач кафедри захисту інформації Національного університету «Львівська політехніка».

Костюк Ю. В., Складанний П.М.

Захист інформації в комп'ютерних системах та мережах. Частина І: підручник / Ю.В. Костюк, П.М. Складанний. – Київ : Київський столичний університет імені Бориса Грінченка, 2026. – 401 с.

У підручнику розглянуто широкий спектр питань, що стосуються забезпечення безпеки комп'ютерних систем та мереж. Особлива увага приділена основам інформаційної безпеки, міжнародним та національним стандартам, правовим аспектам і політикам захисту даних. Детально описано класифікацію кіберзагроз та шкідливого програмного забезпечення, моделі атак та сучасні методи протидії, включно з аналізом соціальної інженерії та інструментів обфускації. У підручнику систематизовано знання щодо захисту комп'ютерних мереж на різних рівнях стеку протоколів, розглянуто практичні механізми та протоколи безпеки, а також виклики, пов'язані з хмарними обчисленнями та SDN-мережами. Значну увагу приділено організаційним і технічним засобам забезпечення інформаційної безпеки: системам контролю доступу, міжмережевим екранам, IDS/IPS, SIEM та SOAR-рішенням, а також інструментам захисту кінцевих точок. Окремий розділ присвячено криптографічним методам захисту, включно з симетричними та асиметричними алгоритмами, цифровими підписами, хеш-функціями та сучасними протоколами захищеного обміну даними. Розкрито перспективи розвитку квантової та гомоморфної криптографії. У підручнику також розглянуто питання організаційного менеджменту інформаційної безпеки, управління ризиками, аудиту та формування корпоративної політики, а також сучасні виклики захисту персональних даних і приватності. Висвітлено аспекти безпеки мобільних і бездротових систем, Інтернету речей, а також критичної інфраструктури. Підручник стане корисним для фахівців з кібербезпеки, студентів і всіх, хто прагнуть системно оволодіти сучасними знаннями та практиками у сфері кіберзахисту.

УДК 004.056:004.7(075.8)

© Костюк Ю. В., Складанний П.М. 2026
© Київський столичний університет імені
Бориса Грінченка, 2026

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ	6
ВСТУП.....	9
РОЗДІЛ 1	11
ОСНОВИ ЗАХИСТУ ІНФОРМАЦІЇ В КОМП'ЮТЕРНИХ СИСТЕМАХ ТА МЕРЕЖАХ: КОНЦЕПЦІЇ, ПРИНЦИПИ, СТАНДАРТИ І ПРАВОВЕ РЕГУЛЮВАННЯ	11
1.1. Еволюція інформаційної безпеки: від перших мереж до сучасних кіберзагроз.....	12
1.2. Базові принципи захисту: конфіденційність, цілісність, доступність, аутентичність, невідмовність.....	20
1.3. Міжнародні стандарти та рамкові моделі безпеки (ISO/IEC 27001, NIST CSF, COBIT, DORA, GDPR).....	24
1.4. Національне та міжнародне правове регулювання у сфері кібербезпеки	31
СПИСОК ДЖЕРЕЛ РОЗДІЛУ 1	37
Контрольні питання	42
РОЗДІЛ 2	44
КІБЕРЗАГРОЗИ ТА ШКІДЛИВЕ ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ: КЛАСИФІКАЦІЯ АТАК, ТЕХНІКИ УХИЛЕННЯ, ЛАНЦЮГИ АТАК І СУЧАСНІ ПІДХОДИ ДО ПРОТИДІЇ.....	44
2.1. Класифікація шкідливого програмного забезпечення: віруси, трояни, руткіти, ботнети, ransomware	45
2.2. Ланцюг кібернападу (Cyber Kill Chain) та MITRE ATT&CK як інструменти аналізу загроз	84
2.3. Техніки ухилення та приховування атак (обфускація, поліморфізм, безфайлові атаки)	96
2.4. Методи соціальної інженерії та фішингові кампанії.....	125
2.5. Інструменти аналізу шкідливого ПЗ і технології протидії ..	130
СПИСОК ДЖЕРЕЛ РОЗДІЛУ 2	137
Контрольні питання	142
РОЗДІЛ 3	145
ЗАХИСТ КОМП'ЮТЕРНИХ МЕРЕЖ ВІД АТАК: МОДЕЛІ ЗАГРОЗ, ВРАЗЛИВОСТІ НА РІЗНИХ РІВНЯХ СТЕКУ ПРОТОКОЛІВ ТА СУЧАСНІ МЕРЕЖЕВІ ТЕХНОЛОГІЇ БЕЗПЕКИ.....	145
3.1. Моделі атакуювальників і загроз: від індивідуальних хакерів до державних АРТ-груп.....	146
3.2. Атаки на мережевому рівні: DDoS, ARP-Spoofing, IP-Spoofing.....	151

3.3. Атаки на транспортному рівні: TCP Hijacking, UDP Flooding, SYN Flood	170
3.4. Атаки на прикладному рівні: SQL Injection, XSS, CSRF, Command Injection	187
3.5. Протоколи та механізми захисту мереж: VPN, IPsec, MPLS, Zero Trust Network Access	211
3.6. Виклики безпеки у хмарних та SDN-мережах	222
3.7. Вразливості та експлойти: від помилок конфігурації до Zero-Day атак	228
3.8. Інструменти моніторингу й виявлення атак: IDS/IPS, SIEM, EDR, NDR	250
СПИСОК ДЖЕРЕЛ РОЗДІЛУ 3	258
Контрольні питання	264
РОЗДІЛ 4	266
<i>ІНСТРУМЕНТИ ТА ТЕХНОЛОГІЇ ЗАБЕЗПЕЧЕННЯ ІНФОРМАЦІЙНОЇ БЕЗПЕКИ: СИСТЕМИ КОНТРОЛЮ ДОСТУПУ, БРАНДМАУЕРИ, IDS/IPS, SIEM/SOAR ТА МОНІТОРИНГ ПОДІЙ</i>	
4.1. Моделі контролю доступу (DAC, MAC, RBAC, ABAC, Zero Trust)	267
4.2. Міжмережеві екрани (статичні, динамічні, NGFW) та їх роль у багаторівневому захисті	276
4.3. Системи виявлення та запобігання вторгнень (IDS/IPS, HIDS, NIDS)	283
4.4. Інструменти SIEM та SOAR для централізованого моніторингу і реагування	293
4.5. Засоби захисту кінцевих точок: антивірусні системи, EDR/XDR	305
СПИСОК ДЖЕРЕЛ РОЗДІЛУ 4	320
РОЗДІЛ 5	328
<i>КРИПТОГРАФІЧНІ МЕТОДИ ЗАБЕЗПЕЧЕННЯ БЕЗПЕКИ: АЛГОРИТМИ ШИФРУВАННЯ, ПРОТОКОЛИ ЗАХИЩЕНОГО ОБМІНУ ТА ПЕРСПЕКТИВИ ПОСТКВАНТОВОЇ КРИПТОГРАФІЇ</i>	
5.1. Математичні основи криптографії та їх значення для захисту даних	330
5.2. Симетричні алгоритми шифрування (AES, ChaCha20, Twofish) та їх застосування	333
5.3. Асиметричні алгоритми (RSA, ECC, постквантові методи) у захисті комунікацій	347
5.4. Хеш-функції, цифрові підписи і контроль цілісності інформації	353

5.5. Протоколи безпечного обміну даними (SSL/TLS 1.3, QUIC, IPsec, WireGuard).....	359
5.6. Сучасні напрями розвитку криптографії: квантова та гомоморфна криптографія).....	365
СПИСОК ДЖЕРЕЛ РОЗДІЛУ 5	367
Контрольні питання	372
<i>СПИСОК РЕКОМЕНДОВАНИХ ДЖЕРЕЛ</i>	375

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

APT	– Advanced Persistent Threat (тривала цілеспрямована загроза)
AES	– Advanced Encryption Standard (стандарт симетричного шифрування)
ABAC	– Attribute-Based Access Control (керування доступом на основі атрибутів)
ACL	– Access Control List (список контролю доступу)
API	– Application Programming Interface (програмний інтерфейс взаємодії)
BCP	– Business Continuity Plan (план безперервності бізнесу)
BIOS	– Basic Input/Output System (базова система введення-виведення)
BOTNET	– мережа заражених комп'ютерів, керованих зловмисником
VPN	– Virtual Private Network (віртуальна приватна мережа)
VoIP	– Voice over IP (передача голосу через IP-мережі)
GDPR	– General Data Protection Regulation (загальний регламент ЄС про захист даних)
GUI	– Graphical User Interface (графічний інтерфейс користувача)
DDoS	– Distributed Denial of Service (розподілена атака на відмову в обслуговуванні)
DNS	– Domain Name System (система доменних імен)
DLP	– Data Loss Prevention (запобігання витоку даних)
DORA	– Digital Operational Resilience Act (акт про цифрову операційну стійкість)
ECDSA	– Elliptic Curve Digital Signature Algorithm (алгоритм цифрового підпису на еліптичних кривих)
EDR	– Endpoint Detection and Response (виявлення та реагування на кінцевих точках)
EMV	– електромагнітне випромінювання
Zero Trust	– архітектура «нульової довіри»
ІБ	– інформаційна безпека
ІКС	– інформаційно-комунікаційна система
ІС	– інформаційна система
ІТ	– інформаційні технології
ІоТ	– Інтернет речей (Internet of Things)
ІоЕ	– Інтернет усього (Internet of Everything)

КСЗІ	– комплексна система захисту інформації
КІД	– конфіденційність, цілісність, доступність (CIA)
КЦІАН	– розширена модель безпеки (Confidentiality, Integrity, Availability, Authentication, Non-repudiation)
MAC (address)	– Media Access Control (унікальний апаратний ідентифікатор мережевого інтерфейсу)
MAC (control)	– Mandatory Access Control (примусовий контроль доступу)
MITM	– Man-in-the-Middle (атака «людина посередині»)
MITRE ATT&CK	– база тактик і технік кіберзагроз
NDR	– Network Detection and Response (виявлення та реагування в мережі)
NIST CSF	– National Institute of Standards and Technology Cybersecurity Framework (рамкова модель кібербезпеки NIST)
НСД	– несанкціонований доступ
OSINT	– Open Source Intelligence (розвідка з відкритих джерел)
ОС	– операційна система
ПБ	– політика безпеки
ПЕМВ	– побічне електромагнітне випромінювання
ПЗ	– програмне забезпечення
PKI	– Public Key Infrastructure (інфраструктура відкритих ключів)
RBAC	– Role-Based Access Control (керування доступом на основі ролей)
RaaS	– Ransomware-as-a-Service (програми-вимагачі як сервіс)
RSA	– Rivest–Shamir–Adleman (алгоритм асиметричного шифрування)
SCADA	– Supervisory Control and Data Acquisition (системи диспетчерського керування і збору даних)
SIEM	– Security Information and Event Management (керування інформацією й подіями безпеки)
SOAR	– Security Orchestration, Automation and Response (автоматизація реагування на інциденти безпеки)
SQLi	– SQL Injection (ін'єкція SQL-коду)
TLS	– Transport Layer Security (захист транспортного рівня)
TCP/IP	– Transmission Control Protocol / Internet Protocol (основний стек мережевих протоколів)

ТЗР ПЕМВН	– технічні засоби розвідки побічних електромагнітних випромінювань і наведень
FTP	– File Transfer Protocol (протокол передавання файлів)
FIPS	– Federal Information Processing Standards (федеральні стандарти обробки інформації)
Firewall	– міжмережевий екран
HTTP/HTTPS	– HyperText Transfer Protocol (Secure) (протокол передавання гіпертексту (захищений))
XML	– Extensible Markup Language (розширювана мова розмітки)
ЦОД	– центр обробки даних (data center)
ЦП	– центральний процесор (CPU)
SHA	– Secure Hash Algorithm (алгоритм хешування)
SSH	– Secure Shell (захищене з'єднання)
LAN	– Local Area Network (локальна мережа)
WAN	– Wide Area Network (глобальна мережа)
Wi-Fi	– Wireless Fidelity (бездротова локальна мережа)
XSS	– Cross-Site Scripting (міжсайтове виконання скриптів)

ВСТУП

Проблематика захисту інформації в комп'ютерних системах та мережах є однією з ключових у сучасному інформаційному суспільстві. Зростаюча складність архітектури обчислювальних систем, інтеграція локальних і глобальних мереж, розвиток хмарних технологій та поява нових класів кіберзагроз вимагають від фахівців ґрунтовного розуміння теоретичних і методологічних основ побудови систем безпеки. Саме ці питання розглядаються у підручнику «Захист інформації в комп'ютерних системах та мережах», який має на меті надати читачеві цілісну картину сучасних концепцій, принципів та засобів забезпечення інформаційної безпеки.

Частина I. Теоретичні та методологічні основи захисту інформації в комп'ютерних системах та мережах присвячена формуванню базових знань, необхідних для подальшого практичного опанування предмету. У цій частині розкриваються фундаментальні принципи інформаційної безпеки — конфіденційність, цілісність, доступність, автентичність та невідомість, — а також еволюція підходів до захисту даних від перших комп'ютерних мереж до сучасних кіберзагроз. Особлива увага приділяється міжнародним стандартам і рамковим моделям (ISO/IEC 27001, NIST CSF, COBIT, DORA, GDPR), національному та міжнародному правовому регулюванню у сфері кіберзахисту, а також ролі політики інформаційної безпеки в управлінні підприємством.

У першій частині також подано систематизований аналіз кіберзагроз і шкідливого програмного забезпечення, методів соціальної інженерії та приховування атак, а також сучасних інструментів їх виявлення і протидії. Окремо розглянуто питання захисту комп'ютерних мереж від атак на різних рівнях стеку протоколів, сучасні підходи до побудови мережевої безпеки, а також виклики, пов'язані з хмарними технологіями та програмно-конфігурованими мережами. Завершальний блок частини присвячено криптографічним методам, які становлять основу захисту інформації в комп'ютерних системах та мережах, охоплюючи як класичні, так і перспективні постквантові підходи.

Матеріали підручника будуть корисними не лише для студентів спеціальностей «Комп'ютерна інженерія», «Кібербезпека та захист інформації», «Інженерія програмного забезпечення», й для здобувачів інших спеціальностей, таких як, «Системний аналіз», «Комп'ютерні науки» та ін. Крім того, підручник стане у пригоді аспірантам і здобувачам ступеня доктор філософії з технічних наук, які досліджують методи захисту

інформації, що базуються на аналізі ризиків, управлінні безпекою інформаційних, комунікаційних та комп'ютерних систем, а також комп'ютерних мереж.

Головною метою посібника є надання студентам базових знань і методологічних засад у сфері захисту інформації в комп'ютерних системах та мережах, формування розуміння ключових принципів побудови багаторівневих систем безпеки, а також оволодіння практичними навичками застосування сучасних технологій кіберзахисту. Особливу увагу приділено поєднанню теоретичних положень із прикладними аспектами, що дозволяє майбутнім фахівцям не лише засвоїти фундаментальні концепції, а й навчитися впроваджувати їх у реальних умовах функціонування інформаційних та комунікаційних систем.

Для зміцнення національної безпеки України питання кіберзахисту набувають особливої ваги, адже стійкість державних інституцій, критичної інфраструктури та бізнес-сектору безпосередньо залежить від ефективності системи захисту інформації. Підручник «Захист інформації в комп'ютерних системах та мережах» покликаний сприяти підготовці висококваліфікованих фахівців, здатних аналізувати й нейтралізувати загрози, розробляти та впроваджувати комплексні заходи інформаційної безпеки, забезпечуючи як інтереси конкретних підприємств, так і загальнодержавні пріоритети.

Таким чином, видання орієнтоване на формування у студентів і слухачів цілісного уявлення про сучасні підходи до кіберзахисту, що є необхідною умовою не лише професійного розвитку майбутніх спеціалістів, а й укріплення цифрового суверенітету та національної безпеки України.

Дослідження проведено в рамках реалізації науково-дослідної теми «Методи та моделі забезпечення кібербезпеки інформаційних систем переробки інформації та функціональної безпеки програмно-технічних комплексів управління критичної інфраструктури» (реєстраційний номер 0122U200483 від 06.07.2022).

РОЗДІЛ 1

ОСНОВИ ЗАХИСТУ ІНФОРМАЦІЇ В КОМП'ЮТЕРНИХ СИСТЕМАХ ТА МЕРЕЖАХ: КОНЦЕПЦІЇ, ПРИНЦИПИ, СТАНДАРТИ І ПРАВОВЕ РЕГУЛЮВАННЯ



- 1.1. Еволюція інформаційної безпеки: від перших мереж до сучасних кіберзагроз***
- 1.2. Базові принципи захисту: конфіденційність, цілісність, доступність, автентичність, невідмовність***
- 1.3. Міжнародні стандарти та рамкові моделі безпеки (ISO/IEC 27001, NIST CSF, COBIT, DORA, GDPR)***
- 1.4. Національне та міжнародне правове регулювання у сфері кібербезпеки***
- 1.5. Роль політики інформаційної безпеки в управлінні підприємством***

Стрімкий розвиток інформаційних технологій та глобальної цифрової інфраструктури зумовив безпрецедентну залежність суспільства, бізнесу та держави від комп'ютерних систем і мереж. Дані стали одним із найцінніших ресурсів сучасності, а їх захист — критичною умовою стабільного функціонування підприємств, урядових структур і навіть повсякденного життя окремих користувачів. Зростання обсягів інформації, її інтеграція у хмарні середовища, мобільні та бездротові платформи, а також розвиток технологій штучного інтелекту відкривають нові можливості, але одночасно створюють численні ризики та виклики у сфері кібербезпеки.

Захист інформації в комп'ютерних системах та мережах є багатовимірною сферою, що поєднує технічні, організаційні, правові та управлінські аспекти. Його основою виступають ключові принципи — конфіденційність, цілісність, доступність, автентичність і невідмовність, які формують фундамент будь-якої системи безпеки. Разом з тим, у сучасному середовищі цього фундаменту недостатньо: ефективний захист має базуватися на міжнародних стандартах, перевірених практиках і правових нормах, які регламентують використання технологій та встановлюють відповідальність за порушення безпеки.

У розділі розглядаються базові концепції та принципи інформаційної безпеки, еволюція уявлень про захист даних, а також ключові стандарти і нормативно-правові акти, що визначають політику кіберзахисту в глобальному та національному масштабах. Особлива увага приділяється гармонізації правових вимог та технічних підходів до безпеки, адже лише комплексний підхід дозволяє забезпечити стійкість інформаційних систем і довіру користувачів у цифрову епоху.

1.1. ЕВОЛЮЦІЯ ІНФОРМАЦІЙНОЇ БЕЗПЕКИ: ВІД ПЕРШИХ МЕРЕЖ ДО СУЧАСНИХ КІБЕРЗАГРОЗ

Правове забезпечення Інформаційна безпека як сфера знань і практичної діяльності пройшла тривалий шлях розвитку, що тісно пов'язаний з еволюцією обчислювальних технологій і телекомунікацій. Ще задовго до появи комп'ютерів люди намагалися захищати дані від сторонніх. Давні цивілізації застосовували прості криптографічні прийоми: спартанці використовували скіталу, а Юлій Цезар впровадив власний шифр для захисту військових повідомлень. У середньовіччі поширилися методи перестановки та підстановки, які лягли в основу подальших алгоритмів шифрування. У XX столітті еволюція криптографії набула технологічного виміру: створення електромеханічних шифрувальних машин, зокрема знаменитої Enigma, продемонструвало роль математичного захисту у військовій та державній безпеці. Подальший розвиток асиметричних алгоритмів, зокрема RSA та протоколу Диффі–Геллмана, дав змогу забезпечувати надійний обмін ключами та захищати комунікації у відкритих мережах.

Подальший розвиток асиметричних алгоритмів, зокрема RSA та протоколу Диффі–Геллмана, дав змогу забезпечувати надійний обмін ключами та захищати комунікації у відкритих мережах. Ці алгоритми стали основою для формування систем цифрових підписів, що гарантують автентичність і цілісність інформації та унеможливають її підроблення. На їх базі було створено протоколи SSL/TLS, які стали стандартом безпечного веб-трафіку й електронної комерції, а також лягли в основу довіри у цифровому середовищі.

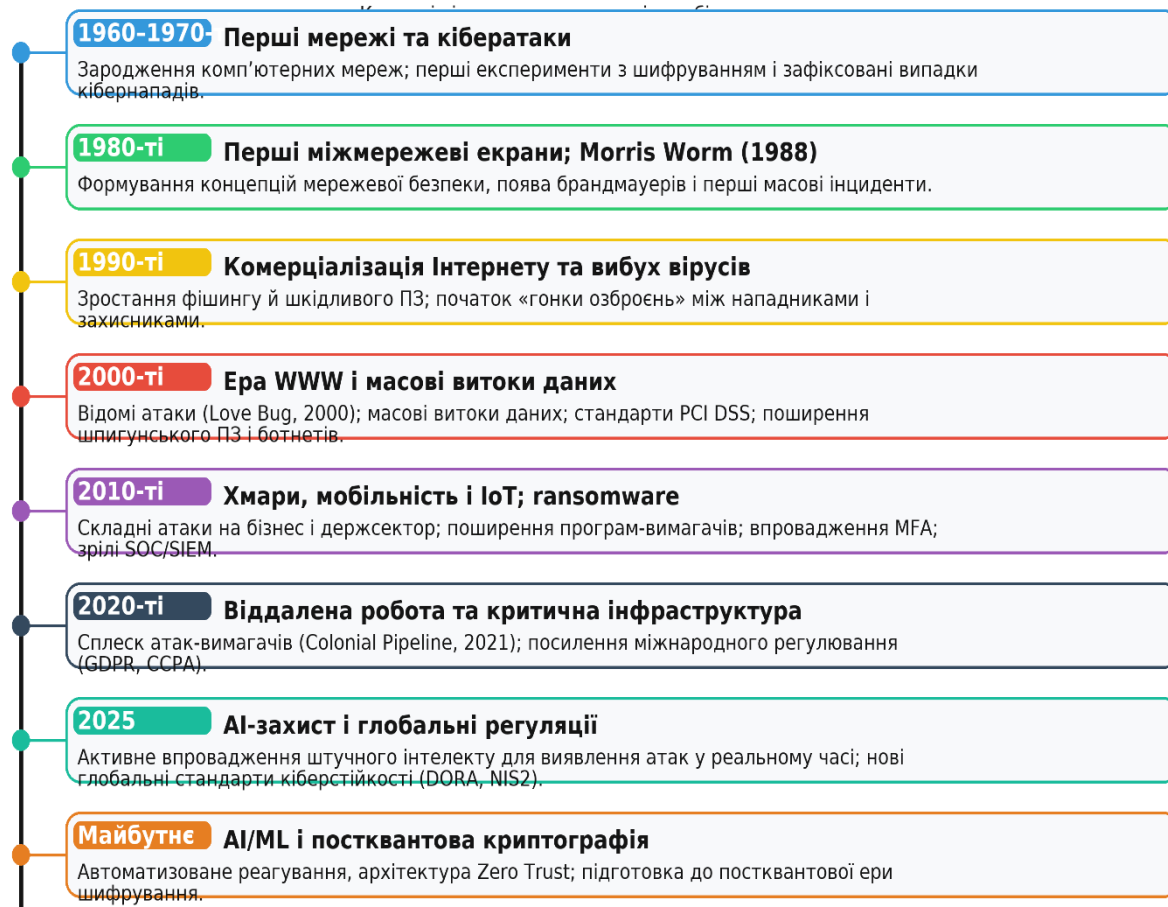


Рис. 1.1. Еволюція кібербезпеки

Джерело: [1-10, 18-23]

Розвиток криптографічних методів сприяв появі багаторівневої аутентифікації та впровадженню інфраструктури відкритих ключів (PKI), яка забезпечила масштабне використання сертифікатів та електронних підписів у державному і корпоративному секторах. Стандарти ISO/IEC 9796 та NIST FIPS 186 закріпили правила застосування цифрових підписів, уніфікувавши практику побудови захищених систем. Водночас практична реалізація протоколів IPsec та SSH продемонструвала, що поєднання асиметричних алгоритмів із симетричними методами дозволяє створювати гібридні системи захисту, які поєднують високу швидкодію з надійним управлінням ключами. Подібні підходи забезпечили безпечну віддалену роботу, захист корпоративних VPN та криптографічне шифрування комунікацій на рівні операційних систем. Асиметрична криптографія стала також невід'ємним компонентом блокчейн-технологій, де використовується для підтвердження транзакцій і створення децентралізованої довіри. У підсумку, еволюція RSA, Диффі–Геллмана та похідних алгоритмів не лише підвищила рівень безпеки даних, але й визначила сучасну архітектуру кіберзахисту,

зробивши її стійкою, масштабованою й придатною до застосування у глобальному середовищі.

З появою перших комп'ютерів у 1950–1960-х роках проблема безпеки спершу зводилася до фізичного контролю доступу. Обчислювальні машини використовувалися переважно у військовій сфері та державних структурах, а їхня вартість і розміри робили ймовірність зовнішнього втручання мінімальною. Проте з розвитком багатокористувацьких операційних систем, таких як Multics і ранні версії UNIX, постала нова загроза — несанкціонований доступ усередині самої системи. Це стало поштовхом до створення формальних моделей контролю доступу, серед яких відомі моделі Белла-Лападули та Біба, які вплинули на подальшу стандартизацію підходів до безпеки комп'ютерних систем.

Модель Белла-Лападули була орієнтована на забезпечення конфіденційності даних. Вона вводила суворі правила доступу, що забороняли користувачам читати інформацію з вищих рівнів секретності («no read up») і записувати дані на нижчі рівні («no write down»). Такий підхід дозволяв мінімізувати ризик витоку конфіденційної інформації та захищав від неконтрольованого поширення секретних даних у багаторівневих системах. На відміну від неї, модель Біба була спрямована на збереження цілісності даних. Основний принцип полягав у тому, що суб'єкти не можуть змінювати інформацію на вищих рівнях цілісності («no write up») і не можуть читати дані з нижчих рівнів («no read down»). Це забезпечувало захист від випадкового або навмисного пошкодження важливої інформації та гарантувало, що критично значущі дані залишаються достовірними.

Обидві моделі заклали основу для сучасних систем контролю доступу та були інтегровані у подальші стандарти інформаційної безпеки. Вони не лише визначили теоретичні засади розмежування прав користувачів, але й стали фундаментом для створення сучасних механізмів багаторівневого захисту, які й сьогодні застосовуються у військових, урядових і корпоративних інформаційних системах.

Ситуація кардинально змінилася із поширенням комп'ютерних мереж у 1970–1980-х роках. Протокол TCP/IP та формування Інтернету надали нові можливості обміну інформацією, але водночас відкрили і нові канали атак. Перші зловмисні втручання мали радше експериментальний характер, проте випадок із «черв'яком Морріса» 1988 року показав, що шкідливий код може спричинити масштабні наслідки для всієї глобальної мережі. Відтоді загроза поширення комп'ютерних вірусів та інших форм шкідливого програмного забезпечення стала актуальною для всіх користувачів.



Рис. 1.2. Ключові інциденти кібербезпеки

Джерело: [1-10, 18-23]

На початку 1990-х з'явилися перші масові вірусні епідемії, які поширювалися через дискети та електронну пошту, що призвело до активного розвитку антивірусної індустрії. Паралельно почали зростати загрози з боку фішингових атак та соціальної інженерії, які використовували людський фактор як слабку ланку в системі безпеки. У цей період великого значення набули міжмережеві екрани (firewalls), які стали базовим інструментом для захисту корпоративних мереж. Наприкінці десятиліття набирали обертів і атаки типу «відмова в обслуговуванні» (DoS), що дозволяли паралізувати цілі інформаційні ресурси. Ці інциденти стимулювали швидкий розвиток комерційного антивірусного програмного забезпечення (AV) і підписно-орієнтованих механізмів виявлення, які почали регулярно оновлюватися для захисту від нових зразків. Поступово з'явилися евристичні та поведінкові методи детекції, що дозволяли виявляти обфусковані або змінені варіанти шкідливого коду, які прості сигнатури пропускали. У відповідь на ескалацію загроз почали впроваджуватися перші системи моніторингу й виявлення вторгнень (IDS — Intrusion Detection

System), практики патч-менеджменту та політики обережного використання зовнішніх носіїв і електронної пошти. Також стартували освітні кампанії для користувачів і були створені координаційні центри реагування (CERT — Computer Emergency Response Team), що заклало організаційну основу для подальшого розвитку національних і міжнародних механізмів кібербезпеки.

У 2000-х роках кіберзлочинність стала більш організованою: зловмисники почали застосовувати ботнети, трояни та шпигунське програмне забезпечення для отримання фінансової вигоди. З'явилися перші масштабні цілеспрямовані атаки на урядові та комерційні структури, які вимагали комплексного підходу до кіберзахисту. Саме тоді міжнародні організації почали активно впроваджувати стандарти інформаційної безпеки, такі як ISO/IEC 27001, що заклали основу сучасних систем менеджменту безпеки інформації. Водночас у відповідь на загострення кіберзагроз сформувалася ціла індустрія кібербезпеки: з'явилися спеціалізовані Центри операцій безпеки (Security Operations Center, SOC), провайдери керованих послуг з безпеки (Managed Security Service Providers, MSSP) та комерційні платформи для моніторингу й відновлення. Активізувалася міжнародна співпраця — були створені команди реагування на комп'ютерні інциденти (Computer Emergency Response Teams / Computer Security Incident Response Teams, CERT/CSIRT) і механізми обміну threat-intelligence (розвідданими про загрози) між державами, галузевими об'єднаннями та провайдерами. На технічному рівні засоби захисту еволюціонували від простих антивірусів до складних стеків таких рішень, як виявлення і реагування на кінцевих точках (Endpoint Detection and Response, EDR), виявлення і реагування в мережі (Network Detection and Response, NDR) та системи керування інформацією й подіями безпеки (Security Information and Event Management, SIEM). Ці трансформації заклали практичну основу для сучасних проактивних підходів до запобігання, виявлення й реагування на інциденти, підвищивши координацію між технічними, організаційними та правовими механізмами захисту.

З початку 2010-х років основним викликом стали програми-вимагачі (ransomware), атаки на хмарні сервіси та інфраструктуру критично важливих об'єктів. Для боротьби з ними поширилися багатофакторна аутентифікація, системи виявлення вторгнень (IDS/IPS) і комплексні рішення SIEM для моніторингу інцидентів. З появою моделі Ransomware-as-a-Service (RaaS) та практик подвійного шантажу зловмисники значно масштабували й комерціалізували операції, зробивши атаки більш організованими і прицільними. Атаки на ланцюги постачання програмного забезпечення та численні інциденти через хмарні misconfiguration підкреслили критичну потребу в управлінні ризиком постачальників (third-party risk), впровадженні Software Bill of Materials (SBOM) та постійній перевірці конфігурацій за допомогою Cloud Security Posture Management (CSPM). У відповідь організації посилили заходи: незмінні

(immutable) бекапи з регулярним тестуванням відновлення, жорсткі політики управління ідентичністю та доступом (Identity and Access Management — IAM), мікросегментація та впровадження Zero Trust-підходу для зменшення площі атаки. Ключовими залишаються операційні практики — відпрацьовані IR-playbook-и, регулярні tabletop-вправи, обмін threat-intelligence і суворе дотримання регуляторних вимог щодо повідомлення про інциденти, що дозволяє скоротити Time To Detect (TTD) і Time To Contain (TTC).

У 2020-х роках відчутно зросла роль штучного інтелекту як у створенні, так і в протидії кіберзагрозам. З'явилися нові виклики, пов'язані з використанням алгоритмів машинного навчання для персоналізованих атак, а також зростає занепокоєння щодо майбутніх загроз, пов'язаних із квантовими обчисленнями. Таким чином, від епохи експериментальних атак до сучасних комплексних сценаріїв кіберзлочинності пройдено шлях, що визначив сучасну парадигму захисту інформації та потребу в глобальній координації зусиль у сфері кібербезпеки.

Штучний інтелект використовується для автоматизації збору розвідданих, генерації більш правдоподібних фішингових повідомлень та оптимізації експлойт-ланцюгів, що підвищує масштаб і ефективність атак. Водночас організації відповідають впровадженням AI-грунтованих систем захисту — від поведінкової аналітики та прогнозної детекції аномалій до автоматизованих SOAR-процедур, що скорочують час реагування. Зростання ролі ML також породжує нові ризики: отруєння навчальних даних, атаки на моделі (model evasion, model extraction) і потребу в прозорих, верифікованих алгоритмах. Паралельно з'являється питання впровадження квантозахисних криптопротоколів, оскільки розвиток квантових обчислень ставить під сумнів стійкість традиційних алгоритмів шифрування. Отже, майбутня парадигма безпеки вимагатиме поєднання технічних інновацій (квантозахист, AI-рішення), регуляторних ініціатив і міжнародної координації для забезпечення надійного та етичного використання нових технологій у кіберпросторі.

У 1990-ті роки з комерціалізацією Інтернету розпочався період стрімкого зростання кіберзлочинності. Віруси Melissa та ILOVEYOU, розвиток троянських програм і ботнетів показали, що атаки можуть мати глобальний характер і завдавати мільйонних збитків. Початкова хвиля кібератак швидко перетворилася на професійну індустрію з розвиненим «ринком» інструментів — експлойт-кітів, дропперів і комерційних сервісів, що забезпечували доступ до шкідливих можливостей навіть непрофесіоналам. У результаті з'явилися фінансово мотивовані сімейства шкідливого ПЗ (банківські трояни, скімінг-модулі, а згодом і програми-вимагачі), які поставили перед організаціями прямі економічні загрози. Ботнети стали інфраструктурою цієї економіки, використовуючись для розсилок, DDoS-кампаній і «оренди» злочинної потужності на підпільних ринках.

Паралельно еволюціонували і цілеспрямовані атаки — АРТ-групи, що поєднували шпигунство, саботаж та промислове втручання, підвищили складність і винахідливість загроз. У відповідь на це розгорнувся розвиток захисних технологій і процедур: від сигнатурних антивірусів до EDR/NDR, SOC-операцій та міжнародної координації у сфері розвідки загроз і правоохоронних takedown-операцій. У цей час сформувалася індустрія антивірусного програмного забезпечення, а міжмережеві екрани стали базовим компонентом корпоративної безпеки. Зростання обсягів фінансових транзакцій та онлайн-сервісів зумовило появу нових методів захисту й одночасно — більш витончених атак.

На початку XXI століття масове поширення ширококутового Інтернету, мобільних технологій та Wi-Fi спричинило глобалізацію загроз. Кіберзлочинність набула організованого характеру, а кіберзагрози вийшли за межі комерційних сфер. Атака на енергосистему Естонії у 2007 році продемонструвала, що кібервплив може паралізувати критичну інфраструктуру цілої держави. Це стало відправною точкою для розуміння кіберпростору як нового театру бойових дій, де застосовуються методи кібертероризму та кібервійни. Цей випадок підкреслив кілька важливих уроків: по-перше, сучасні кібероперації здатні завдати економічно та соціально відчутної шкоди поза межами ІТ-інфраструктури, уражаючи енерго- та транспортні системи, банківську інфраструктуру й комунікації. По-друге, атака на Естонію показала важливість швидкої координації між державою і приватним сектором — постраждалі сервіси належали як державним, так і комерційним операторам, і відновлення вимагало скоординованих зусиль. Третій урок — проблема атрибуції: визначення джерела та мотивів атак ускладнює правове й політичне реагування, що стимулювало розвиток технік цифрової форензики й міжнародної співпраці з обміну розвідданими. На тлі цих подій посилилися інвестиції у захист критичної інфраструктури — впровадження стандартів безпеки для SCADA/ICS-систем, сегментація мереж, резервні канали зв'язку і плани безперервності бізнесу стали обов'язковими елементами стратегії стійкості. Також зріс запит на формування міжнародних норм поведінки в кіберпросторі, участь у навчаннях і спільних вправах (tabletop, red-team), а також створення національних та міждержавних CERT/CSIRT-центрів. Загалом, події початку XXI століття перевели кібербезпеку з переважно технічної проблеми у сферу національної безпеки й політики, зробивши необхідним системний підхід, що поєднує технічні, організаційні та правові заходи.

Друга декада XXI століття принесла вибуховий розвиток хмарних технологій, мобільних застосунків та Інтернету речей. Водночас зросла кількість витоків персональних даних: масштабні інциденти з Equifax, Yahoo та Facebook продемонстрували вразливість навіть найбільших корпорацій. Нові ризики виникли у багатокористувацькому середовищі хмар, де неправильні

налаштування чи вразливості віртуалізації призводять до масових компрометацій. Особливу небезпеку становить використання уразливих IoT-пристроїв, які часто стають частиною ботнетів, застосовуваних для масових DDoS-атак. Ці події стимулювали перехід до нових підходів у безпеці — зокрема до Zero Trust-моделей, жорсткішого управління ідентичностями та доступом (IAM), шифрування даних у спокої та при передачі, а також широкого застосування засобів виявлення й усунення неправильних конфігурацій у хмарі (CSPM) та інструментів сканування вразливостей (SAST/DAST) для CI/CD-пайплайнів. Зросла роль управління постачальницькими ризиками: аудит ланцюгів постачання ПЗ, вимоги до постачальників щодо безпечних оновлень і прозорих SBOM стали частиною корпоративних політик. Щоб протидіяти загрозам з боку IoT, організації впроваджують сегментацію мереж, обмеження життєвого циклу пристроїв, централізоване управління оновленнями та суворіші вимоги до аутентифікації пристроїв. Також посилилися вимоги до резервування й тестування планів аварійного відновлення, оскільки в умовах багатокористувацьких хмар відновлення даних і сервісів стало критичним для збереження бізнес-операцій. У відповідь на масштабні витоки даних прийшли посилені регуляторні ініціативи й підвищена увага до захисту приватності (підвищення штрафів, обов'язкове повідомлення про інциденти), що змушує компанії інтегрувати безпеку й приватність у життєвий цикл розробки продуктів. Загалом, друга декада показала, що інновації у ІТ створюють як нові переваги, так і нові загрози, і що ефективна стратегія повинна поєднувати технологічні, організаційні та регуляторні заходи.

Сучасний етап розвитку інформаційної безпеки позначений зростанням складності загроз та використанням новітніх технологій як у захисних, так і у зловмисних цілях. Штучний інтелект і машинне навчання дозволяють розробляти ефективні системи виявлення аномалій, прогнозування атак і автоматизованого реагування, проте одночасно вони ж застосовуються кіберзлочинцями для створення генеративних фішингових атак і обходу систем захисту. Квантові обчислення, що перебувають у стадії розвитку, становлять загрозу для сучасних криптографічних алгоритмів, стимулюючи активні дослідження у сфері постквантової криптографії.

Крім технічних викликів, сучасна парадигма кібербезпеки підкреслює важливість управління даними й прозорості моделей: інтерпретовані та верифіковані ML-системи знижують ризики помилкових спрацьовувань і роблять можливою кореляцію сигналів у масштабі підприємства. Одночасно зростає загроза від атак на самі моделі — отруєння навчальних даних, атаки на стійкість моделей (evasion) та викрадення інтелектуальної власності (model extraction) вимагають нових процедур захисту життєвого циклу ML. Розвиток постквантової криптографії та стандартизація алгоритмів захисту від квантових атак стають пріоритетом для критичних секторів, де довговічність конфіденційності даних

особливо важлива. Практичні стратегії включають поетапну міграцію до квантозахисчених схем, гібридні криптосистеми й ретельне управління ключами разом із регулярним аудитом криптографічних практик. Не менш важливими залишаються організаційні заходи: підвищення кваліфікації персоналу, інтеграція DevSecOps та розробка етичних і регуляторних рамок для застосування AI в безпеці. На міжнародному рівні необхідна координація стандартів, обмін threat intelligence і узгоджені підходи до реагування на інциденти, оскільки атаки все частіше мають транскордонний характер. У підсумку, поєднання технологічних інновацій, прозорих процедур і глобальної співпраці визначатиме здатність суспільства протистояти еволюції кіберзагроз.

Таким чином, еволюція інформаційної безпеки відображає шлях розвитку цифрової цивілізації: від простих шифрів античності до комплексних стратегій кіберзахисту в умовах глобальних мереж. Кожен історичний етап породжував нові виклики й вимагав нових рішень, формуючи багаторівневу систему знань і технологій, без якої сучасний світ не може функціонувати. Сьогодні інформаційна безпека сприймається не як одноразовий захід, а як безперервний процес адаптації до динамічних загроз, де технічні, організаційні та правові підходи взаємодоповнюють одне одного, забезпечуючи стійкість суспільства в умовах цифрової епохи.

1.2. БАЗОВІ ПРИНЦИПИ ЗАХИСТУ: КОНФІДЕНЦІЙНІСТЬ, ЦІЛІСНІСТЬ, ДОСТУПНІСТЬ, АУТЕНТИЧНІСТЬ, НЕВІДМОВНІСТЬ

Сучасні системи захисту інформації базуються на фундаментальних принципах, які визначають вимоги до безпеки будь-якої інформаційно-комунікаційної інфраструктури. Ці принципи формують так звану CIAAN-модель (Confidentiality, Integrity, Availability, Authentication, Non-repudiation), що є основою як міжнародних стандартів (ISO/IEC 27001, ISO/IEC 15408), так і практичних методів побудови захищених систем.

У межах цієї моделі конфіденційність гарантує збереження даних від несанкціонованого доступу, цілісність забезпечує їхню достовірність і захист від змін, доступність гарантує отримання інформації у потрібний час. Аутентифікація підтверджує справжність користувачів і систем, а невідмовність запобігає можливості заперечення фактів виконаних дій. Разом ці п'ять принципів створюють єдиний каркас, що дозволяє будувати політики безпеки на всіх рівнях — від технічних засобів і протоколів до управлінських процедур. Впровадження цих принципів вимагає вибудови багаторівневої стратегії, що включає шифрування, контроль доступу, резервне копіювання, моніторинг і відпрацьовані процедури реагування на інциденти. Постійний

аудит, тестування (наприклад, пентестинг і відновлення з резервних копій) та навчання персоналу забезпечують адаптацію каркасу до нових загроз і підтримують його ефективність.

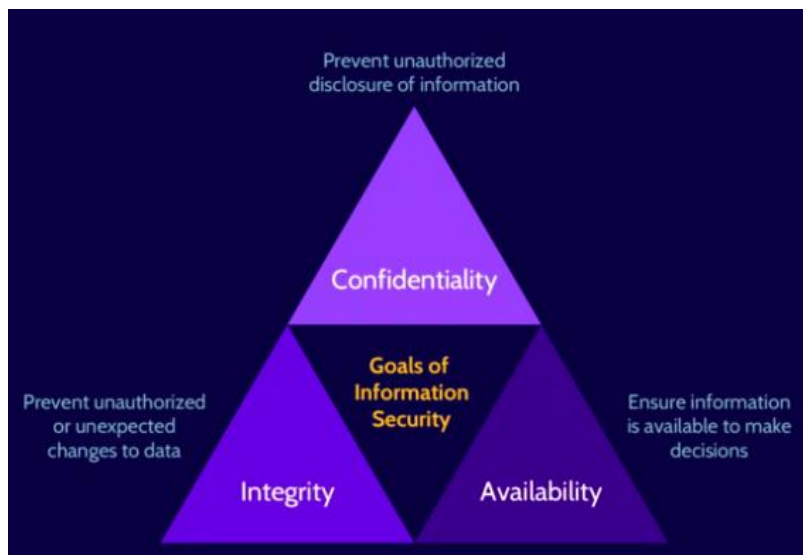


Рис. 1.3. Базові цілі інформаційної безпеки (модель CIA: конфіденційність, цілісність, доступність)

Джерело: [1-19, 23-30]

Практична цінність CIAAN-моделі полягає в тому, що вона дає універсальний підхід до оцінки ризиків і визначення заходів захисту незалежно від масштабу системи — від корпоративної мережі до національної критичної інфраструктури. У сучасних умовах цифрової трансформації та зростання кількості кібератак ця модель виступає основою для розробки нових стандартів, нормативних вимог і програмних рішень у сфері інформаційної безпеки.

Конфіденційність (Confidentiality) означає збереження інформації від несанкціонованого доступу. Тільки уповноважені користувачі мають отримувати доступ до даних відповідно до своїх прав і повноважень. Для забезпечення конфіденційності застосовуються криптографічні методи (симетричне та асиметричне шифрування, протоколи TLS, VPN), політики управління доступом (RBAC, ABAC), сегментація мереж і технології багатофакторної аутентифікації. Конфіденційність є критично важливою для захисту персональних даних, комерційної таємниці, державної інформації з обмеженим доступом. Втрата конфіденційності може призвести до масштабних витоків даних, фінансових збитків та репутаційних втрат для підприємства. У багатьох країнах законодавство (наприклад, GDPR у ЄС) передбачає серйозні санкції за неналежний захист конфіденційної інформації. Сучасні підходи до захисту базуються на концепції «найменших привілеїв» (principle of least

privilege), яка мінімізує ризики несанкціонованого доступу. Крім того, важливу роль відіграє навчання персоналу, адже людський фактор часто є найслабшою ланкою в системі збереження конфіденційності.

Цілісність (Integrity) означає захист інформації від несанкціонованих змін або знищення. Інформація має залишатися достовірною й повною протягом усього життєвого циклу. Для контролю цілісності використовуються хеш-функції (SHA-2, SHA-3), цифрові підписи, механізми контролю версій, журнали аудиту. Порушення цілісності може відбутися як унаслідок зовнішніх атак (внесення шкідливого коду, модифікація файлів), так і через внутрішні помилки (збої програмного забезпечення, недбалість персоналу). Забезпечення цілісності є особливо важливим у фінансових системах, медичних даних і критично важливих інфраструктурах. Втрати цілісності даних можуть призвести до прийняття хибних рішень, що особливо небезпечно у сферах захисту, медицини чи банківської діяльності. Для зменшення ризиків впроваджуються системи контролю доступу, багаторівневі перевірки і процедури резервного копіювання. У міжнародних стандартах, таких як ISO/IEC 27001, цілісність визначається як один із ключових аспектів інформаційної безпеки поряд із конфіденційністю та доступністю. Сучасні організації також застосовують автоматизовані засоби моніторингу змін у даних, що дозволяє своєчасно виявляти і блокувати спроби несанкціонованого втручання.

Доступність (Availability) означає, що інформація та ресурси мають бути доступними користувачам у момент, коли вони необхідні. Втрата доступності може бути наслідком атак типу «відмова в обслуговуванні» (DoS/DDoS), збоїв апаратного забезпечення, перебоїв у роботі енергосистем чи мережевого обладнання. Для підтримання доступності застосовуються резервування серверів, балансування навантаження, системи відновлення після збоїв (Disaster Recovery, BCP), хмарні сервіси з геореплікацією даних. Висока доступність (High Availability, HA) є критичною умовою для банківських систем, e-commerce, медичних сервісів, телекомунікацій і державних онлайн-послуг. Недостатня доступність може призвести до зупинки бізнес-процесів, втрати клієнтів та значних фінансових збитків. З цієї причини підприємства все частіше інвестують у хмарні технології та розподілені центри обробки даних, які забезпечують безперервність роботи навіть у разі аварій. У критичних інфраструктурах, таких як енергетика чи транспорт, доступність розглядається як питання національної безпеки. Для забезпечення цього принципу застосовують не лише технічні рішення, а й організаційні заходи — чіткі регламенти реагування на інциденти, навчання персоналу та регулярне тестування планів безперервності бізнесу.

Аутентифікація (Authentication) означає перевірку достовірності суб'єкта або об'єкта доступу. Це гарантує, що користувач або система є саме тим, за кого себе видає. Аутентифікація забезпечується механізмами багатофакторної перевірки (пароль, токен, біометрія), цифровими сертифікатами (PKI), протоколами Kerberos і OAuth 2.0. Наявність надійної аутентифікації є фундаментальною передумовою побудови Zero Trust-архітектур, де кожна дія користувача перевіряється незалежно від місця його підключення. Слабка або неправильно налаштована аутентифікація відкриває шлях до компрометації облікових записів і несанкціонованого доступу до критичних систем. Сучасні тенденції вимагають поступового відходу від класичних паролів на користь біометричних методів, апаратних токенів і passwordless-рішень. Велике значення має й контекстна аутентифікація, коли для перевірки враховується не лише введений пароль чи відбиток пальця, але й місце розташування користувача, його пристрій або навіть поведінкові характеристики. У поєднанні з механізмами управління ідентичностями (IAM) аутентифікація створює основу для комплексної системи контролю доступу, яка здатна адаптуватися до нових кіберзагроз і підтримувати високий рівень безпеки.

Невідмовність (Non-repudiation) означає неможливість відмови від здійснених дій або переданих даних. Це гарантує, що відправник не може заперечити факт відправлення повідомлення, а отримувач — факт його отримання. Для цього застосовуються електронні цифрові підписи (ECDSA, RSA), сертифікати, протоколи безпечного електронного документообігу. Принцип невідмовності є ключовим у юридично значущих транзакціях, фінансових операціях, судовій практиці та електронному врядуванні. Реалізація цього принципу підвищує рівень довіри між сторонами, адже кожна дія в системі набуває підтвердженого статусу. Недотримання вимог невідмовності може призвести до судових спорів, шахрайства та фінансових втрат. Сучасні технології, такі як блокчейн, значно розширюють можливості забезпечення невідмовності завдяки децентралізованій та незмінній природі записів. Крім того, невідмовність тісно пов'язана з іншими принципами безпеки, зокрема з цілісністю та аутентифікацією, утворюючи єдину систему гарантій захищеності інформаційних процесів.

Таким чином, конфіденційність, цілісність, доступність, аутентифікація та невідмовність формують ядро сучасної інформаційної безпеки. Вони взаємодоповнюють одне одного і створюють основу для побудови комплексних систем захисту, здатних протистояти сучасним кіберзагрозам і забезпечувати стійкість інформаційних процесів у цифрову епоху.

1.3. МІЖНАРОДНІ СТАНДАРТИ ТА РАМКОВІ МОДЕЛІ БЕЗПЕКИ (ISO/IEC 27001, NIST CSF, COBIT, DORA, GDPR)

Міжнародні стандарти та рамкові моделі безпеки відіграють ключову роль у формуванні сучасної політики захисту інформації, оскільки вони визначають єдині правила, принципи й процедури для підприємств різних галузей і масштабів. Їхня мета полягає у створенні системного підходу до управління ризиками, забезпечення конфіденційності, цілісності та доступності даних, а також дотримання правових і регуляторних вимог. Серед найбільш поширених та впливових стандартів і моделей варто виділити ISO/IEC 27001, NIST Cybersecurity Framework (CSF), COBIT, DORA та GDPR. Вони забезпечують уніфікацію підходів до кіберзахисту, що дозволяє підприємствам різних країн ефективно співпрацювати та обмінюватися інформацією. Виконання вимог цих документів підвищує рівень довіри між бізнесом, державними структурами та користувачами. Крім того, стандарти й рамкові моделі формують основу для розвитку національних стратегій кібербезпеки та гармонізації міжнародного законодавства.

ISO/IEC 27001 є міжнародним стандартом, що визначає вимоги до створення, впровадження та вдосконалення системи управління інформаційною безпекою (ISMS). Він охоплює всі аспекти захисту інформації — від організаційних політик і контролю доступу до технічних заходів і внутрішніх аудитів. Використання ISO/IEC 27001 дозволяє підприємствам продемонструвати відповідність найкращим практикам, а також зміцнити довіру партнерів і клієнтів. Стандарт має ризик-орієнтований підхід, що вимагає від організацій систематично виявляти, оцінювати й управляти інформаційними ризиками. Він інтегрується з іншими управлінськими системами, такими як ISO 9001 (якість) чи ISO 22301 (безперервність бізнесу), створюючи єдину екосистему корпоративного управління. Значну увагу приділено вимогам до документування процесів, що забезпечує прозорість і контрольованість системи. Наявність сертифікації ISO/IEC 27001 часто стає конкурентною перевагою на ринку, особливо в сферах, де критично важливим є захист даних клієнтів. Крім того, він враховує як технічні аспекти (криптографія, управління інцидентами), так і людський фактор (навчання персоналу, управління доступами). Стандарт постійно оновлюється, що дозволяє йому відповідати новим викликам кібербезпеки та враховувати глобальні тенденції розвитку інформаційних технологій.



Рис. 1.4. Основні принципи стандарту ISO/IEC 27001
Джерело: [1-19, 23-30]

NIST Cybersecurity Framework (CSF), розроблений Національним інститутом стандартів і технологій США, пропонує п'ять ключових функцій кібербезпеки: ідентифікація, захист, виявлення, реагування та відновлення. Ця рамка широко застосовується як у державному, так і в приватному секторі для побудови адаптивних стратегій протидії кіберзагрозам. Її гнучкість дозволяє інтегрувати NIST CSF у будь-які організаційні процеси управління ризиками.

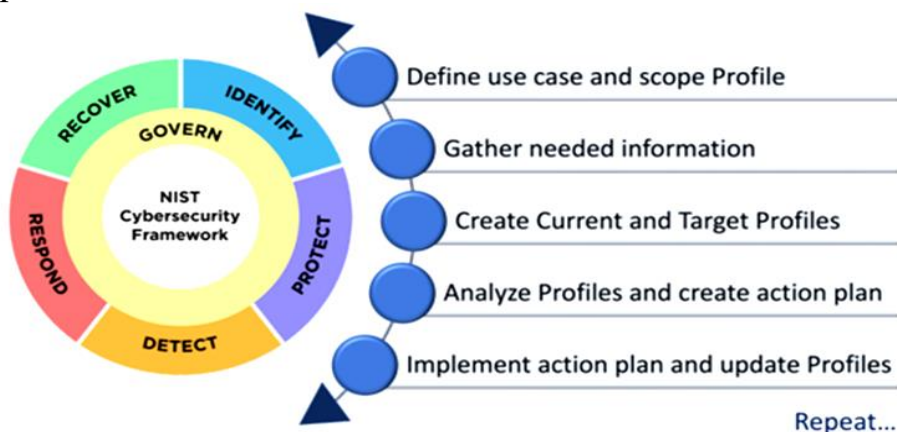


Рис. 1.5. Цикл побудови профілів безпеки за NIST CSF (Define → Gather → Create → Analyze → Implement)

Джерело: [1-19, 23-30, 58-61]

Фреймворк не є жорстким стандартом, а радше методологією, яку можна адаптувати до специфіки підприємства, його масштабу та галузі. Він надає спільну мову для обговорення кіберризиків між технічними фахівцями, керівництвом і регуляторами, що підвищує прозорість управлінських рішень.

Function	Category	Category Identifier
Govern (GV)	Organizational Context	GV.OC
	Risk Management Strategy	GV.RM
	Cybersecurity Supply Chain Risk Management	GV.SC
	Roles, Responsibilities, and Authorities	GV.RR
	Policies, Processes, and Procedures	GV.PO
	Oversight	GV.OV
Identify (ID)	Asset Management	ID.AM
	Risk Assessment	ID.RA
	Improvement	ID.IM
Protect (PR)	Identity Management, Authentication, and Access Control	PR.AA
	Awareness and Training	PR.AT
	Data Security	PR.DS
	Platform Security	PR.PS
	Technology Infrastructure Resilience	PR.IR
Detect (DE)	Continuous Monitoring	DE.CM
	Adverse Event Analysis	DE.AE
Respond (RS)	Incident Management	RS.MA
	Incident Analysis	RS.AN
	Incident Response Reporting and Communication	RS.CO
	Incident Mitigation	RS.MI
Recover (RC)	Incident Recovery Plan Execution	RC.RP
	Incident Recovery Communication	RC.CO



Рис. 1.6. NIST Cybersecurity Framework 2.0: співвідношення функцій із категоріями та ідентифікаторами
Джерело: [1-19, 23-30, 58-61]

Використання NIST CSF допомагає організаціям визначати пріоритети інвестицій у безпеку, балансуючи між витратами й рівнем захищеності. Значна увага приділяється безперервному вдосконаленню: після проходження циклу заходів організація аналізує досягнуті результати й формує плани для підвищення зрілості своєї системи безпеки. Крім того, NIST CSF є сумісним з іншими міжнародними стандартами, такими як ISO/IEC 27001, що дозволяє підприємствам ефективно поєднувати різні підходи. У результаті ця модель стала де-факто глобальним еталоном для побудови сучасних стратегій кіберзахисту.

COBIT (Control Objectives for Information and Related Technologies) — це комплексна модель управління ІТ і безпекою, що допомагає організаціям досягати бізнес-цілей за рахунок ефективного використання інформаційних технологій. COBIT фокусується на відповідності, управлінні ризиками, оптимізації ресурсів і підвищенні прозорості процесів. У сфері кібербезпеки він часто використовується як інструмент аудиту та оцінки зрілості управлінських практик. Основна цінність COBIT полягає в тому, що він

забезпечує єдину методологію для узгодження стратегічних бізнес-цілей із завданнями ІТ-підрозділів.

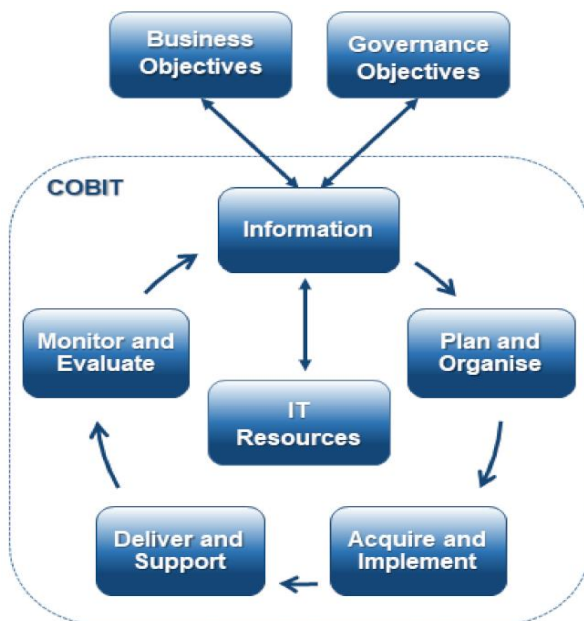


Рис. 1.7. COBIT framework: Plan, Acquire, Deliver, Monitor — інформаційний цикл і ІТ-ресурси

Джерело: [1-19, 23-30, 58-61]

Рамкова модель включає набір процесів, показників і контрольних цілей, які дозволяють вимірювати ефективність управління та впроваджувати постійне вдосконалення. COBIT активно використовується як база для побудови внутрішніх і зовнішніх аудитів, адже він охоплює не лише технічні, а й організаційні аспекти управління інформаційними ресурсами. Важливою особливістю є його орієнтація на принципи корпоративного управління, що робить його зручним інструментом для керівництва в ухваленні рішень. Крім того, COBIT можна інтегрувати з іншими міжнародними стандартами, зокрема ISO/IEC 27001 і NIST CSF, що підсилює ефективність системи безпеки. У сучасних умовах він набуває особливого значення як інструмент оцінки зрілості кібербезпеки підприємства та визначення пріоритетних напрямів її розвитку.

DORA (Digital Operational Resilience Act) — новий регламент Європейського Союзу, спрямований на зміцнення операційної стійкості фінансових установ. Він встановлює вимоги до управління ІКТ-ризиками, кіберзахисту, тестування стійкості систем і взаємодії з постачальниками критичних цифрових послуг. DORA зобов'язує фінансовий сектор створювати ефективні механізми для запобігання та швидкого реагування на кіберінциденти, що має критичне значення для захисту економічної

стабільності ЄС. Регламент охоплює широкий спектр установ — від банків і страхових компаній до постачальників платіжних послуг і криптовалютних платформ. Його ключова особливість полягає у підвищеній увазі до управління ризиками в ланцюгу постачання цифрових технологій, адже збої у роботі сторонніх провайдерів можуть мати критичні наслідки для всієї фінансової системи.

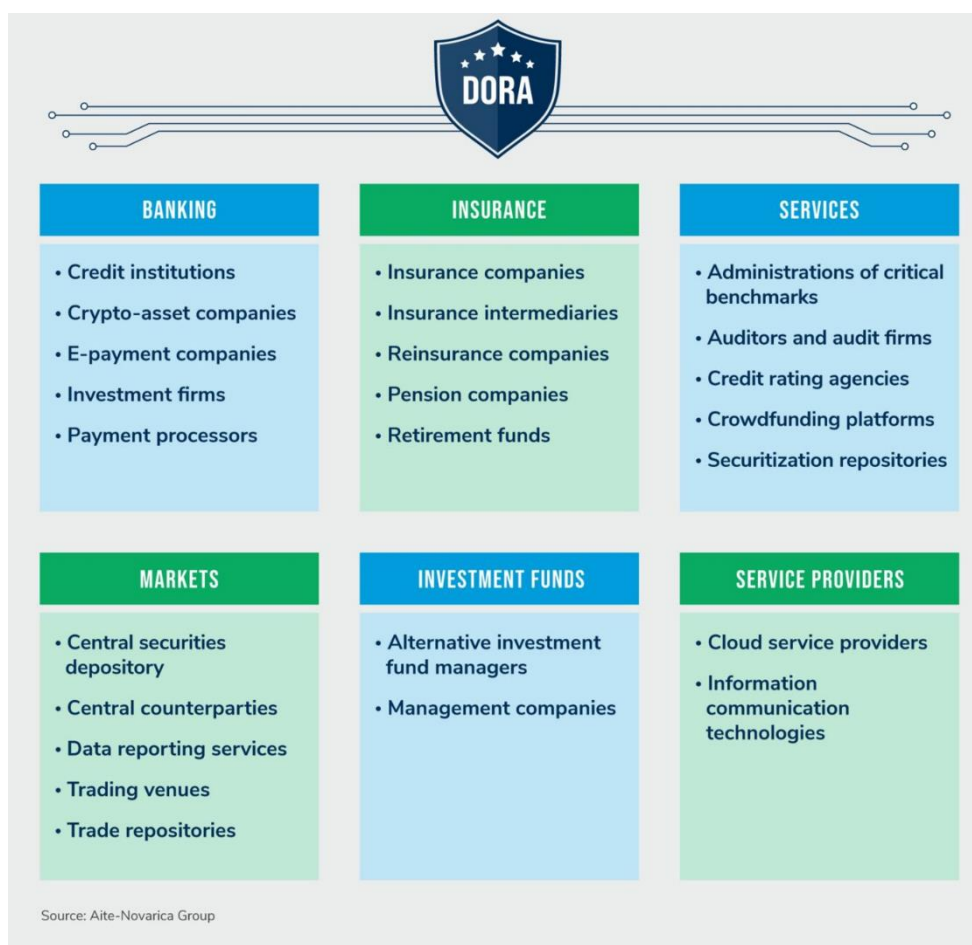


Рис. 1.8. DORA: категорії підпадаючих суб'єктів (банки, страхові компанії, постачальники послуг тощо)

Джерело: [1-19, 23-30, 58-61]

DORA вимагає регулярного проведення тестувань кіберстійкості, включно з симуляціями реальних атак (threat-led penetration testing), що дозволяє оцінити готовність систем до надзвичайних ситуацій. Також регламент встановлює стандартизовані вимоги до звітування про кіберінциденти та їх комунікації з регуляторами. У результаті він сприяє гармонізації вимог безпеки в межах ЄС та створює єдиний підхід до кіберзахисту фінансового сектору. Прийняття DORA стало важливим кроком у

формуванні сучасної європейської моделі цифрової безпеки, яка може слугувати прикладом для інших регіонів світу.

GDPR (General Data Protection Regulation) — загальний регламент ЄС із захисту персональних даних, що встановлює суворі правила їх збирання, обробки, зберігання та передачі. Він надає громадянам широкі права на контроль своїх даних і передбачає значні штрафи для підприємств у разі порушень. GDPR став еталонним законодавчим актом, який вплинув на розвиток національних і корпоративних політик із захисту персональної інформації в усьому світі.



Рис. 1.9. Принципи GDPR: законність, обмеження цілей, мінімізація даних, точність, обмеження зберігання, цілісність і конфіденційність, підзвітність
Джерело: [1-19, 23-30, 58-61]

Основні положення регламенту передбачають право на доступ до власних даних, право на виправлення або видалення («право бути забутим»), а також право на перенесення даних між сервісами. Компанії зобов'язані впроваджувати принципи «privacy by design» та «privacy by default», тобто враховувати вимоги захисту приватності вже на етапі проектування систем і сервісів. Важливу роль відіграє інститут Data Protection Officer (DPO) — спеціаліста, який відповідає за відповідність діяльності організації вимогам GDPR. Регламент має екстериторіальну дію: він поширюється на будь-які компанії у світі, якщо вони обробляють персональні дані громадян ЄС. Високі штрафи — до 20 млн євро або 4% річного обороту компанії — роблять GDPR одним із найсуворіших механізмів регулювання у сфері інформаційної безпеки. У глобальному масштабі він став каталізатором для ухвалення подібних законів у різних країнах, формуючи єдину тенденцію до підвищення стандартів захисту персональних даних.

Таблиця 1.1. Порівняння міжнародних стандартів та рамок моделей кібербезпеки (ISO/IEC 27001, NIST CSF, COBIT, DORA, GDPR)

Стандарт / Модель	Основна мета	Сфера застосування	Ключові принципи та функції	Особливості / Вимоги
ISO/IEC 27001	Створення та впровадження системи управління інформаційною безпекою (ISMS)	Універсальна — для підприємств будь-якої галузі та масштабу	Ризик-орієнтований підхід, контроль доступу, внутрішні аудити, політики безпеки	Сертифікація підтверджує відповідність міжнародним практикам; інтегрується з ISO 9001, ISO 22301
NIST CSF	Управління кіберризиками та підвищення стійкості систем	США та глобально (добровільна основа)	5 функцій: ідентифікація, захист, виявлення, реагування, відновлення	Гнучка рамка, адаптивна до будь-якої організації; сумісність з ISO/IEC 27001
COBIT	Управління ІТ-ресурсами й забезпечення відповідності бізнес-цілям	Корпоративний сектор, аудит ІТ	Контрольні цілі, процеси, метрики ефективності	Використовується для оцінки зрілості; інтегрується з ISO та NIST; фокус на корпоративному управлінні
DORA	Забезпечення цифрової операційної стійкості фінансового сектору	Фінансові установи ЄС: банки, страхові, платіжні системи, криптоплатформи	Управління ІКТ-ризиками, тестування стійкості, управління інцидентами	Обов'язковий регламент ЄС; включає threat-led penetration testing, контроль постачальників
GDPR	Захист персональних даних і приватності громадян ЄС	Усі компанії, що обробляють дані громадян ЄС (глобальна дія)	Privacy by design, privacy by default, права користувачів (доступ, видалення, перенесення)	Суворі штрафи (до 20 млн євро або 4% обороту); створення ролі DPO; глобальний вплив на законодавство

Табл. 1.1. показує, що ISO/IEC 27001 і NIST CSF забезпечують універсальний підхід до управління інформаційною безпекою, COBIT зосереджений на IT-управлінні та аудиті, DORA має вузьку фінансову спрямованість, а GDPR регулює саме персональні дані й приватність.

У сукупності ці стандарти та рамкові моделі створюють багаторівневу систему, яка дозволяє підприємствам забезпечити відповідність регуляторним вимогам, мінімізувати ризики та ефективно протидіяти сучасним кіберзагрозам. Вони також формують основу для інтеграції технічних і організаційних заходів, що забезпечують стійкість інформаційних систем навіть в умовах постійно зростаючого рівня складності атак.

1.4. НАЦІОНАЛЬНЕ ТА МІЖНАРОДНЕ ПРАВОВЕ РЕГУЛЮВАННЯ У СФЕРІ КІБЕРБЕЗПЕКИ

Правове регулювання у сфері кібербезпеки є одним із ключових інструментів забезпечення захищеності інформаційного простору, адже воно визначає правила функціонування цифрових систем, встановлює вимоги до суб'єктів господарювання та державних органів, а також окреслює відповідальність за порушення у сфері інформаційної безпеки. На національному рівні країни ухвалюють закони, що регламентують основи кіберзахисту, діяльність уповноважених органів і порядок реагування на інциденти.

Важливим аспектом є захист критичної інформаційної інфраструктури, для якої встановлюються спеціальні вимоги щодо стійкості та резервування систем. Багато держав також вводять обов'язок повідомляти про кіберінциденти у визначені терміни, що сприяє оперативному реагуванню та координації дій. Правові норми передбачають інтеграцію технічних, організаційних і кадрових заходів безпеки, формуючи комплексний підхід до захисту. Крім того, розвиток законодавства у цій сфері відбувається у тісному зв'язку з міжнародними стандартами та угодами, що забезпечує гармонізацію національних стратегій із глобальними практиками.



Рис. 1.10. Системна схема національного та міжнародного правового регулювання кібербезпеки

Джерело: [1-19, 23]

Наприклад, в Україні головним нормативним актом є Закон «Про основні засади забезпечення кібербезпеки України» (2017 р.), який визначає принципи національної політики у цій сфері, встановлює повноваження Держспецзв'язку, Кіберполіції та Національного координаційного центру кібербезпеки, а також зобов'язує критичні об'єкти інфраструктури впроваджувати належні заходи захисту. Додатково діє Закон «Про захист персональних даних», який гармонізується з вимогами GDPR і передбачає права громадян на контроль за використанням їхніх персональних даних. У країнах Європейського Союзу діють аналогічні національні закони, що узгоджуються з Директивою NIS (Network and Information Security Directive) та оновленою її редакцією NIS2, яка розширює перелік операторів критичної інфраструктури і посилює вимоги до управління кіберризиками.

У межах NIS2 особливу увагу приділено обов'язковому звітуванню про кіберінциденти та підвищенню рівня відповідальності керівництва компаній за їхній захист. Держави-члени ЄС повинні створювати національні органи, відповідальні за моніторинг і координацію у сфері кібербезпеки, що сприяє гармонізації політик усередині Союзу. В Україні у цьому контексті активно розвивається співпраця з ЄС для інтеграції у спільний кіберпростір та поступового наближення законодавства до європейських норм. Крім того, правове поле вимагає від підприємств і державних установ не лише впроваджувати технічні засоби захисту, а й проводити регулярні аудити

безпеки та навчання персоналу. Подібний підхід дозволяє забезпечити комплексність системи захисту, охоплюючи як технологічні, так і організаційні складові. Таким чином, національне законодавство України та країн ЄС рухається у напрямку гармонізації з міжнародними вимогами, що є важливою умовою для ефективної боротьби з глобальними кіберзагрозами.

На міжнародному рівні ключову роль відіграють багатосторонні угоди та конвенції, що визначають спільні підходи до боротьби з кіберзлочинністю та захисту даних. Найбільш відомим документом є Будапештська конвенція про кіберзлочинність (2001 р.), яка стала першим міжнародним договором, що криміналізує несанкціонований доступ до систем, втручання в роботу мереж і розповсюдження шкідливого програмного забезпечення, а також передбачає механізми співпраці між правоохоронними органами різних країн. Значний вплив на глобальну практику справив Регламент ЄС GDPR (2018 р.), який встановив нові стандарти захисту персональних даних, обов'язкові для компаній у всьому світі, якщо вони працюють із громадянами ЄС. У США одним із ключових документів є Cybersecurity Information Sharing Act (CISA, 2015), що сприяє обміну інформацією про кіберзагрози між урядом і приватними компаніями, тоді як Національний інститут стандартів і технологій (NIST) розробив Cybersecurity Framework, який став практичним орієнтиром для багатьох держав і корпорацій. Крім того, міжнародні організації — ООН, ОБСЄ, НАТО — просувають ініціативи зі створення правил відповідальної поведінки держав у кіберпросторі, а також реалізують програми з підвищення кіберстійкості країн-партнерів.

Важливо відзначити, що такі ініціативи сприяють формуванню довіри між державами та створенню механізмів колективної відповіді на масштабні кібератаки. Наприклад, у межах Європейського Союзу активно діють агентства ENISA та CERT-EU, які координують реагування на кіберінциденти та поширюють найкращі практики серед держав-членів. У глобальному вимірі дедалі більшої актуальності набувають питання створення універсальних правил поведінки в кіберпросторі, які б поєднували принципи міжнародного права з особливостями цифрового середовища. Водночас дискусії щодо суверенітету в кіберпросторі та права держав на захист своїх цифрових кордонів залишаються відкритими, що підкреслює складність вироблення єдиних міжнародних стандартів. Таким чином, міжнародне правове регулювання поступово еволюціонує від фрагментарних угод до більш комплексної системи норм і правил, яка здатна забезпечити глобальну безпеку в умовах зростання кіберзагроз.

Сучасним викликом у сфері правового регулювання є надзвичайно швидкий розвиток технологій, що часто випереджає темпи оновлення нормативних актів. Поява штучного інтелекту, квантових обчислень і нових

типів кібератак потребує адаптації законодавства, створення більш гнучких норм та узгодження правил на міжнародному рівні. Перспективним напрямом є гармонізація законодавства різних країн, адже глобальний характер кіберзлочинності вимагає єдиних підходів до захисту даних, обміну інформацією та переслідування правопорушників. Таким чином, поєднання національного й міжнародного правового регулювання формує багаторівневу систему, що забезпечує правові основи кібербезпеки та створює підґрунтя для розвитку безпечного цифрового середовища у глобалізованому світі.

1.5. Роль політики інформаційної безпеки в управлінні підприємством

Політика інформаційної безпеки (ПІБ) є фундаментальним елементом системи управління сучасним підприємством, оскільки вона формує базові правила, принципи й стандарти поведінки з інформаційними ресурсами. Її головна мета полягає у визначенні єдиного підходу до захисту даних, незалежно від рівня працівника, технологій або підрозділів, що їх обробляють. ПІБ забезпечує узгодженість дій між бізнес-цілями та вимогами кіберзахисту, допомагаючи уникнути хаотичного та фрагментарного впровадження заходів безпеки. Вона визначає сферу дії (активи, процеси, дані), ролі та відповідальність учасників, а також процедури схвалення винятків і періодичного перегляду. ПІБ включає класифікацію даних, правила доступу, вимоги до шифрування, резервного копіювання та зберігання журналів аудиту. Невід'ємною частиною політики є програми навчання персоналу та регулярні тренування з реагування на інциденти, що зменшують людський фактор. Для контролю виконання встановлюються метрики та KPI (наприклад, час виявлення/реагування, відсоток покриття MFA, рівень актуальності патчів) і проводяться внутрішні та зовнішні аудити. Політика поширюється на взаємодію з підрядниками й постачальниками, вимагаючи договорів про рівень безпеки (SLAs) та оцінок ризиків ланцюга постачання. Вона має життєвий цикл «розробка — затвердження — впровадження — моніторинг — вдосконалення», що забезпечує її актуальність у динамічному середовищі загроз.

По-перше, політика виступає нормативною основою управління ризиками, визначаючи ключові активи підприємства, загрози для них і прийнятні методи захисту. Вона інтегрує механізми оцінки ризиків у стратегічне планування, що дозволяє керівництву ухвалювати рішення, ґрунтуючись на балансі між рівнем безпеки та витратами. Вона також чітко фіксує відповідальності та ролі — хто відповідає за безпеку, процедури ескалації інцидентів і механізми звітності. Політика визначає вимоги до технічних контролів (наприклад, MFA — багатофакторна аутентифікація, шифрування, тестовані резервні копії) і критерії відповідності нормативам та

стандартам (наприклад, ISO/IEC 27001 — міжнародний стандарт з управління інформаційною безпекою, GDPR — Загальний регламент захисту даних), що спрощує аудит і взаємодію з регуляторами. Крім того, політики мають містити набір вимірюваних KPI (ключових показників ефективності) — наприклад, TTD (Time To Detect — час виявлення), TTC (Time To Contain — час утримання), відсоток покриття технік MITRE ATT&CK (Adversarial Tactics, Techniques and Common Knowledge) — і механізми їх періодичного перегляду залежно від змін ризик-ландшафту. Нарешті, ефективна політика поєднується з практичними процесами (інструкції, playbook-и для реагування), регулярним навчанням персоналу та технічними засобами, формуючи циклічну систему управління безпекою з постійним удосконаленням.

По-друге, ПІБ виконує організаційну функцію, закріплюючи відповідальність і ролі працівників у процесах забезпечення безпеки. Це включає визначення обов'язків ІТ-відділу, керівників підрозділів і кінцевих користувачів, а також запровадження процедур реагування на інциденти, розслідування порушень і проведення внутрішніх аудитів. Політика інформаційної безпеки (ПІБ) встановлює чіткі ролі й відповідальність — від власників систем і Chief Information Security Officer (CISO) до адміністраторів та кінцевих користувачів — і регламентує повноваження, права прийняття рішень і процедури ескалації. Вона також документує вимоги до розподілу обов'язків (segregation of duties), контролю доступу, управління привілеями та процедур надання й відкликання доступу, щоб зменшити ризик зловживань і внутрішніх помилок. У межах ПІБ прописуються стандарти роботи ІТ-підрозділу (change management, patch-management, резервне копіювання), механізми координації під час інцидентів, а також ролі forensic-команди й юридичного відділу у зборі доказів та взаємодії з регуляторними органами. Регулярні внутрішні аудити, навчання персоналу, відпрацювання інцидент-сценаріїв (tabletop, технічні вправи) і вимірювальні KPI забезпечують, що ПІБ є живим інструментом управління ризиком, а не формальною процедурою.

По-третє, політика інформаційної безпеки є інструментом дотримання законодавчих і нормативних вимог. Вона узгоджується з міжнародними стандартами (ISO/IEC 27001, NIST CSF, COBIT), галузевими регламентами (PCI DSS, HIPAA, DORA) і національним законодавством (наприклад, «Про основні засади забезпечення кібербезпеки України», «Про захист персональних даних»). Таким чином, ПІБ допомагає підприємству уникати штрафів, судових позовів та репутаційних втрат. Вона також встановлює процедури документування й збереження доказів відповідності — журнали доступу, результати аудитів і звіти про ризики — що полегшує проходження перевірок і демонстрацію належної обачності (due diligence). ПІБ визначає обов'язкові строки повідомлення регуляторів і постраждалих осіб у разі інциденту, а також

формати звітності, що зменшує правові ризики і пришвидшує реакцію на запити контролюючих органів. Наявність сертифікацій (наприклад, відповідності ISO/IEC 27001 або проходження оцінки за NIST CSF) полегшує укладання контрактів з партнерами і замовниками, для яких важливі доказово підтверджені заходи безпеки. Нарешті, ППБ має бути живим документом: регулярний перегляд у зв'язку зі змінами в нормативному полі, результатами аудитів і еволюцією технічних загроз гарантує, що захист залишається актуальним і відповідним вимогам.

По-четверте, ППБ сприяє формуванню корпоративної культури безпеки, оскільки визначає правила користування електронною поштою, Інтернетом, мобільними пристроями, системами віддаленого доступу. Працівники отримують чіткі інструкції щодо захисту паролів, використання багатофакторної аутентифікації, поводження з конфіденційною інформацією. Вона має задавати регулярні навчальні програми й практичні тренінги (включно з фішинг-симуляціями) та вимірювати їхню ефективність через показники, наприклад click-rate на тренувальні листи і time-to-report від користувачів. ППБ повинна закріплювати безпечні процедури для BYOD (Bring Your Own Device — використання власних пристроїв), правил роботи з конфіденційною інформацією і вимог до захищених конфігурацій пристроїв і додатків. Важливо також створити «non-punitive» культуру повідомлення про інциденти, тобто чіткі канали звітування і заохочення за раннє повідомлення замість покарань за помилки. Підтримка з боку керівництва, регулярний моніторинг комплаєнсу і включення навчання в систему оцінювання персоналу перетворюють політику з формального документа на живу практику, що реально знижує ризики.

По-п'яте, добре розроблена політика створює основу для безперервності бізнесу. У ній закладаються механізми резервного копіювання, відновлення після аварій, планування дій у надзвичайних ситуаціях, що забезпечує стійкість підприємства навіть за умов масштабних кібератак або технічних збоїв. Вона також формалізує вимоги до визначення цілей відновлення — Recovery Time Objective (RTO) та Recovery Point Objective (RPO) — і вимагає проведення Business Impact Analysis (аналізу впливу на бізнес) для пріоритизації сервісів і даних. Політика закладає вимоги до архітектури резервного копіювання (включно з immutable — незмінними копіями), регулярного тестування бекапів і відновлення з них у контрольованому середовищі, а також наявності альтернативних (жорстко ізольованих або cold/hot) майданчиків для критичних сервісів. Вона визначає процедури комунікації під час кризових ситуацій (внутрішні канали, повідомлення клієнтам і регуляторам), порядок ескалації та перелік відповідальних осіб, що дозволяє уникнути хаосу в перші години інциденту. Нарешті, ППБ вимагає регулярного тестування планів безперервності

(tablettop, full-scale drills) і внесення корективів на підставі результатів, що гарантує оперативну придатність процедур у реальних умовах.

Отже, політика інформаційної безпеки є не лише технічним чи адміністративним документом, а стратегічним інструментом управління, який дозволяє підприємству ефективно функціонувати в умовах цифрової економіки, зберігаючи довіру клієнтів, партнерів і державних органів.

СПИСОК ДЖЕРЕЛ РОЗДІЛУ 1

1. Костюк Ю.В., Складанний П.М., Бебешко Б.Т., Хорольська К.В., Рзаєва С.Л., Ворохоб М.В. (2025) Безпека інформаційно-комунікаційних систем. Київський університет імені Бориса Грінченка, Україна, Київ. - 1017 с.
2. Костюк Ю.В., Складанний П.М., Гулак Г.М., Бебешко Б.Т., Хорольська К.В., Рзаєва С.Л. (2025) Системи захисту інформації. Київський університет імені Бориса Грінченка, Україна, Київ. - 886с.
3. Kostiuk, Yu. V., Skladannyi, P. M., Bebeshko, B. T., Khorolska, K. V., Rzaieva, S. L., & Vorokhob, M. V. (2025). Information and communication systems security. [Textbook] Kyiv: Borys Grinchenko Kyiv Metropolitan University.
4. Kostiuk, Yu. V., Skladannyi, P. M., Hulak, H. M., Bebeshko, B. T., Khorolska, K. V., & Rzaieva, S. L. (2025). Information security systems. [Textbook] Kyiv: Borys Grinchenko Kyiv Metropolitan University.
5. Гулак Г. М., Жильцов О. Б., Киричок Р. В., Коршун Н. В., Складанний П. М. Інформаційна та кібернетична безпека підприємства : підруч. / Г. М. Гулак, О. Б. Жильцов, Р. В. Киричок, Н. В. Коршун, П. М. Складанний – Львів : Видавець Марченко Т. В., 2023. – 370 с.
6. Шевченко, С., Жданова, Ю., & Гаркушенко, А. (2025). Когнітивний підхід в інформаційній та кібербезпеці. Електронне фахове наукове видання «Кібербезпека: освіта, наука, техніка», 1(29), 854–866. <https://doi.org/10.28925/2663-4023.2025.29.945>
7. Чернігівський, І., & Крючкова, Л. (2025). Тестування нейромережових моделей для вирішення задачі виявлення заражених ПК на базі цифрових слідів. Електронне фахове наукове видання «Кібербезпека: освіта, наука, техніка», 1(29), 800–817. <https://doi.org/10.28925/2663-4023.2025.29.941>
8. Романюк , В., & Платоненко, А. (2025). Аналіз якості генераторів рожевого шуму. Електронне фахове наукове видання «Кібербезпека: освіта, наука, техніка», 1(29), 789–799. <https://doi.org/10.28925/2663-4023.2025.29.940>
9. Соболенко , І., & Платоненко, А. (2025). Автоматизоване виявлення аномалій у трафіку корпоративних бездротових мереж за допомогою Python: методи, реалізація та оцінка ефективності. Електронне фахове наукове видання

«Кібербезпека: освіта, наука, техніка», 1(29), 777–788.
<https://doi.org/10.28925/2663-4023.2025.29.939>

10. Соколов, В. (2025). Технологія відслідковування переміщення абонентів територією підприємства критичної інфраструктури. Електронне фахове наукове видання «Кібербезпека: освіта, наука, техніка», 1(29), 207–222.
<https://doi.org/10.28925/2663-4023.2025.29.920>

11. Складанний, П., Гулак, Г., & Корнієць, В. (2025). Коаліційний підхід до управління кібербезпекою інформаційних систем що застосовують хмарні технології. Кібербезпека: освіта, наука, техніка, 4(28), 8–25.
<https://doi.org/10.28925/2663-4023.2025.27.825>

12. Довженко, Н., Іваніченко, Є., Аушева, Н., Шевчук, Ю., & Луковський, Т. (2025). Дослідження архітектури дата-центрів з інтеграцією IoT-компонентів для забезпечення енергоефективності та кіберстійкості. Кібербезпека: освіта, наука, техніка, 4(28), 547–564.
<https://doi.org/10.28925/2663-4023.2025.28.835>

13. Методичні рекомендації до виконання курсової роботи з дисципліни «Захист інформації в інформаційно-комунікаційних системах» для студентів спеціальності 125 Кібербезпека та захист інформації, освітньої програми 125.00.01 Безпека інформаційних і комунікаційних систем / Укладачі: Костюк Ю.В., Складанний П.М., Рзаєва С.Л. Київ: КСУБГ, 2025. 70 с.

14. Шевченко, С., Жданова, Ю., Складанний, П., & Іщук, М. (2025). Створення блокчейн-платформи для електронного голосування. Кібербезпека: освіта, наука, техніка, 4(28), 701–714. <https://doi.org/10.28925/2663-4023.2025.28.860>

15. Основи баз даних : навчальний посібник / Рзаєва С.Л., Машкіна І.В., Складанний П.М., Костюк Ю.В., Рзаєв Д.О., Красюк Ю.М – Київ : Київський столичний університет імені Бориса Грінченка, 2025. – 319 с.

16. Крючкова, Л., & Шандрук, М. (2025). Методи протидії в радіонавігаційних конфліктах. Кібербезпека: освіта, наука, техніка, 4(28), 766–780. <https://doi.org/10.28925/2663-4023.2025.28.863>

17. Цехмейстер, Р., Платоненко, А., Ворохоб, М., Черевик, В., & Семеняка, С. (2025). Дослідження методів забезпечення інформаційної безпеки у віртуальному середовищі. Кібербезпека: освіта, наука, техніка, 3(27), 63–71.
<https://doi.org/10.28925/2663-4023.2025.27.703>

18. Kostiuk, Y., Skladannyi, P., Korshun, N., Bebeshko, B., & Khorolska, K. Integrated protection strategies and adaptive resource distribution for secure video streaming over a Bluetooth network. Workshop on Cybersecurity Providing in Information and Telecommunication Systems II, (CPITS-II 2024) October 26, 2024. Aachen: CEUR, 2024. Vol. 3826. P. 129–138. ISSN: 1613-0073.

19. Олійник, Я., Платоненко, А., Черевик, В., Ворохоб, М., & Шевчук, Ю. (2025). Методи захисту інформації в технологіях IoT. Кібербезпека: освіта, наука, техніка, 3(27), 100–108. <https://doi.org/10.28925/2663-4023.2025.27.705>
20. Дмитрієнко, К., & Коршун, Н. (2025). Застосування вдосконалених моделей Курамото ДЛІА ідентифікації дезінформації у соціальних мережах. Кібербезпека: освіта, наука, техніка, 3(27), 406–416. <https://doi.org/10.28925/2663-4023.2025.27.754>
21. Kostiuk, Y., Skladannyi, P., Samoilenko, Y., Khorolska, K., Bebeshko, B., Sokolov, V. A system for assessing the interdependencies of information system agents in information security risk management using cognitive maps. Workshop on Third International Conference on Cyber Hygiene & Conflict Management in Global Information Networks, (CH&CMiGIN'24) January 24–27, 2024. Aachen: CEUR, 2024. Vol. 3925. P. 249-264. ISSN 1613-0073.
22. Методичні рекомендації до виконання курсової роботи з дисципліни «Захист інформації в комп'ютерних системах та мережах» для студентів спеціальності 123 Комп'ютерна інженерія, освітньої програми 123.00.01 Комп'ютерна інженерія / Укладачі: Костюк Ю.В., Складанний П.М., Рзаєва С.Л. Київ: КСУБГ, 2025. 82 с.
23. Kostiuk, Y., Skladannyi, P., Sokolov, V., Hulak, H., Korshun, N. Models and algorithms for analyzing information risks during the security audit of personal data information system. Workshop on Third International Conference on Cyber Hygiene & Conflict Management in Global Information Networks, (CH&CMiGIN'24) January 24–27, 2024. Aachen: CEUR, 2024. Vol. 3925. P. 155-171. ISSN 1613-0073.
24. Довженко, Н., Складанний, П., Іваніченко, Є., & Жильцов, О. (2025). Модель динамічної взаємодії безпілотних літальних апаратів із сенсорною мережею для енергоефективного моніторингу. Кібербезпека: освіта, наука, техніка, 3(27), 466–478. <https://doi.org/10.28925/2663-4023.2025.27.766>
25. Mashkina I., Rzaieva S., Kostiuk Y., Mazur N., Brzhevska Z. Cybersecurity in Intelligent Transport Systems: Current Challenges and Solutions. Proceedings of the Workshop on Cybersecurity Providing in Information and Telecommunication Systems (CPITS 2025), 2025. Aachen: CEUR, 2025. Vol. 3991. P. 14-25. ISSN 1613-0073.
26. Гайдук, О., Зверев, В., & Гулак, Г. (2025). Кореляційний аналіз типів кіберінцидентів в контексті інформаційних технологій міжнародних комунікацій. Кібербезпека: освіта, наука, техніка, 3(27), 417–428. <https://doi.org/10.28925/2663-4023.2025.27.757>
27. Kostiuk Y., Skladannyi P., Sokolov V., Zhyltsov O., Ivanichenko Y. Effectiveness of Information Security Control using Audit Logs. Proceedings of the Workshop on Cybersecurity Providing in Information and Telecommunication

Systems (CPITS 2025), 2025. Aachen: CEUR, 2025. Vol. 3991. P. 524-538. ISSN 1613-0073.

28. Kostiuk Y., Skladannyi, P., Sokolov, V., Vorokhob, M. Models and technologies of cognitive agents for decision-making with integration of Artificial Intelligence. Proceedings of the Modern Data Science Technologies Doctoral Consortium (MoDaST 2025). Aachen: CEUR, 2025. Vol. 4005. P. 82–96. ISSN 1613-0073.

29. Чернігівський, І., & Крючкова, Л. (2025). Системний підхід до вирішення задачі захисту інформації в інфокомунікаційній мережі від впливу комп'ютерних вірусів. Кібербезпека: освіта, наука, техніка, 3(27), 572–590. <https://doi.org/10.28925/2663-4023.2025.27.781>

30. Skladannyi P., Kostiuk Y., Rzaieva S., Bebeshko, B., Korshun, N. Adaptive Methods for Embedding Digital Watermarks to Protect Audio and Video Images in Information and Communication Systems. Proceedings of the Workshop Classic, Quantum, and Post-Quantum Cryptography, August 5, 2025, Kyiv, Ukraine (CQPC 2025). Vol. 4016. P. 13–31. ISSN 1613-0073.

31. Чернігівський, І., & Крючкова, Л. (2025). Ефективні рішення для швидкого виявлення скомпрометованих ПК в інфокомунікаційних мережах. Телекомунікаційні та інформаційні технології, 2(87), 24–32. <https://doi.org/10.31673/2412-4338.2025.029875>

32. Kostiuk Y., Skladannyi, P., Khorolska, K., Sokolov, V., Hulak, H. Application of Statistical and Neural Network Algorithms in Steganographic Synthesis and Analysis of Hidden Information in Audio and Graphic Files. Proceedings of the Workshop Classic, Quantum, and Post-Quantum Cryptography, August 5, 2025, Kyiv, Ukraine (CQPC 2025). Vol. 4016. P. 45–65. ISSN 1613-0073.

33. Чернігівський, І., & Крючкова, Л. (2024). Тестування антивірусних рішень для корпоративного сегменту. Безпека інформації, 30(3), 407–413. <https://doi.org/10.18372/2225-5036.30.20362>

34. Skladannyi, P., Kostiuk Y., Khorolska, K., Bebeshko, B., Sokolov, V. Model and methodology for the formation of adaptive security profiles for the protection of wireless networks in the face of dynamic cyber threats. Proceedings of the Workshop Cyber Security and Data Protection, July 31, 2025, Lviv, Ukraine (CSDP 2025). Vol. 4042. P. 17–36. ISSN 1613-0073.

35. Соколов, В. (2025). Забезпечення стійкості безпроводових систем до атак глушіння. Телекомунікаційні та інформаційні технології, 1(86), 50–60. <https://doi.org/10.31673/2412-4338.2025.013623>

36. Kostiuk, Y., Rzaieva S., Khorolska, K., Mazur, N., Korshun, N. Architecture of the software system of confidential access to information resources of computer networks. Proceedings of the Workshop Cyber Security and Data

Protection, July 31, 2025, Lviv, Ukraine (CSDP 2025). Vol. 4042. P. 37–53. ISSN 1613-0073.

37. Корнієць, В., & Складанний, П. (2024). Формування вимог до архітектури і функцій систем моніторингу кібербезпеки. Телекомунікаційні та інформаційні технології, 4(85), 90–96. <https://doi.org/10.31673/2412-4338.2024.040224>

38. Kostiuk, Y., Skladannyi, P., Sokolov, V., Rzaieva S.(2025) Intelligent System for Simulation Modeling and Research of Information Objects. Proceedings of the 1st Workshop Software Engineering and Semantic Technologies (SEST 2025), co-located with the 15th International Scientific and Practical Programming Conference (UkrPROG 2025), May 13–14, 2025, Kyiv, Ukraine, Vol-4053, P. 237-251. ISSN 1613-0073.

39. Агашков, А., Шевченко, С., Бондарчук, А., & Жебка, В. (2024). Шляхи підвищення транспортних операцій за допомогою хмарних логістичних рішень. Телекомунікаційні та інформаційні технології, 4(85), 4–15. <https://doi.org/10.31673/2412-4338.2024.047191>

40. Sokolov, V., Kostiuk, Y., Skladannyi, P., Korshun, N. (2025). Adaptation of Network Traffic Routing Policy to Information Security and Network Protection Requirements. Proceedings of the 13th International Scientific and Practical Conference "Information Control Systems and Technologies", ICST 2025, Vol-4048, P. 397 - 411. ISSN 1613-0073.

41. Бондарчук, А., Складанний, П., Жебка, В., & Стражніков, А. (2024). Оптимізація системи зарядки електромобілів на основі методів дослідження операцій. Телекомунікаційні та інформаційні технології, 3(84), 86–93. <https://doi.org/10.31673/2412-4338.2024.038693>

42. Крючкова, Л., Башкевич, Є., & Куцибала, М. (2024). Програмна реалізація алгоритмів виокремлення контурів зображень об'єктів в інтелектуальних системах відеоспостереження. Телекомунікаційні та інформаційні технології, 2(83), 14–29. <https://doi.org/10.31673/2412-4338.2024.022439>

43. Оксанич, І., Гречанінов, В., Литвинов, В., & Складанний П. (2024). Особливості забезпечення гарантоздатності та кіберстійкості інформаційного обміну в складних умовах. Телекомунікаційні та інформаційні технології, 2(83), 105–113. <https://doi.org/10.31673/2412-4338.2024.022128>

44. Крючкова, Л., Яремчук, Д., & Пазинін, А. (2024). Планування навантаження контейнерів за допомогою лінійного програмування. Телекомунікаційні та інформаційні технології, 2(83), 114 –119. <https://doi.org/10.31673/2412-4338.2024.022935>

45. Цирканюк, Д., & Соколов, В. (2024). Методика розслідування інцидентів інформаційної безпеки. *Кібербезпека: освіта, наука, техніка*, 2(26), 140–154. <https://doi.org/10.28925/2663-4023.2024.26.675>
46. Козачок, В., & Драпатий, М. (2024). Аналіз технології розслідування інцидентів безпеки на об'єктах критичної інфраструктури. *Кібербезпека: освіта, наука, техніка*, 2(26), 374–391. <https://doi.org/10.28925/2663-4023.2024.26.699>
47. Шевченко, С., Жданова, Ю., Складанний, П., & Петренко, Т. (2024). Нечіткі когнітивні карти як інструмент візуалізації сценаріїв реагування на інциденти в системах безпеки. *Кібербезпека: освіта, наука, техніка*, 2(26), 417–429. <https://doi.org/10.28925/2663-4023.2024.26.707>
48. Корнієць, В., & Жданова, Ю. (2024). Методика формування швидкісного імітостійкого шифру багато алфавітної заміни. *Кібербезпека: освіта, наука, техніка*, 2(26), 476–486. <https://doi.org/10.28925/2663-4023.2024.26.723>

КОНТРОЛЬНІ ПИТАННЯ

1. Чому сучасне суспільство стає все більш залежним від комп'ютерних систем і мереж, і які ризики це породжує?
2. Які п'ять принципів складають CIAAN-модель і що означає кожен із них?
3. Наведіть приклади технічних і організаційних заходів, що забезпечують конфіденційність даних.
4. Чим порушення цілісності відрізняється від порушення конфіденційності? Наведіть по одному прикладу кожного.
5. Які рішення застосовують для підвищення доступності сервісів на рівні інфраструктури та процесів?
6. Які механізми аутентифікації згадано в тексті та в чому перевага багатофакторної перевірки?
7. Що таке невідмовність, як вона досягається на практиці і в яких випадках є критичною?
8. У чому історичне значення алгоритмів RSA та Диффі–Геллмана для побудови захищених комунікацій?
9. Поясніть сутність гібридного шифрування і чому воно стало стандартом у протоколах на кшталт TLS, IPsec, SSH.
10. У чому різниця між моделями Белла-Лападули та Біба? Які правила доступу лежать в їх основі?
11. Яке значення мав «черв'як Морріса» (1988) для усвідомлення мережеских загроз?

12. Які тенденції кіберзагроз характерні для 1990-х, 2000-х, 2010-х і 2020-х років відповідно? Наведіть по 1–2 приклади.
13. Що визначає стандарт ISO/IEC 27001 і чому сертифікація за ним створює конкурентну перевагу?
14. Назвіть п'ять функцій NIST CSF і поясніть, чому рамка вважається гнучкою.
15. Яку роль відіграє COBIT у поєднанні бізнес-цілей та ІТ-управління і як його використовують під час аудитів?
16. Які ключові вимоги та сфера дії регламенту DORA, і на які організації він поширюється?
17. Які права суб'єктів даних і які санкції передбачає GDPR? Поясніть принципи privacy by design/default.
18. Які основні елементи національного правового поля кібербезпеки України та як воно узгоджується з NIS/NIS2?
19. Які міжнародні документи та ініціативи формують глобальні правила: Будапештська конвенція, CISA, NIST CSF, ініціативи ООН/НАТО? Яку мету переслідують?
20. Які п'ять управлінських ролей виконує політика інформаційної безпеки в підприємстві (ризики, відповідальність, відповідність, культура, безперервність) і як це впливає на стійкість бізнесу?

РОЗДІЛ 2

КІБЕРЗАГРОЗИ ТА ШКІДЛИВЕ ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ: КЛАСИФІКАЦІЯ АТАК, ТЕХНІКИ УХИЛЕННЯ, ЛАНЦЮГИ АТАК І СУЧАСНІ ПІДХОДИ ДО ПРОТИДІЇ



- 2.1. Класифікація шкідливого програмного забезпечення: віруси, трояни, руткіти, ботнети, ransomware**
- 2.2. Ланцюг кібернападу (Cyber Kill Chain) та MITRE ATT&CK як інструменти аналізу загроз**
- 2.3. Техніки ухилення та приховування атак (обфускація, поліморфізм, безфайлові атаки)**
- 2.4. Методи соціальної інженерії та фішингові кампанії**
- 2.5. Інструменти аналізу шкідливого ПЗ і технології протидії**

Розділ присвячений комплексному аналізу сучасних викликів у сфері інформаційної безпеки. У ньому розглядаються основні види шкідливого програмного забезпечення, включаючи віруси, трояни, руткіти, ботнети та програми-вимагачі, що є найбільш поширеними інструментами кіберзлочинців. Особливу увагу приділено механізмам їх поширення, способам ураження та впливу на конфіденційність, цілісність і доступність даних. Далі представлено аналіз ланцюга кібернападу (Cyber Kill Chain), який демонструє послідовність дій зловмисників від рекогносцировки до експлуатації та підтримання присутності, а також використання моделі MITRE ATT&CK як практичного

інструмента систематизації тактик і технік атак для підвищення ефективності виявлення загроз. У тексті також описані сучасні техніки ухилення та приховування, зокрема обфускація, поліморфізм, метаморфізм та безфайлові атаки, що ускладнюють роботу систем детекції та антивірусного захисту. Значна увага приділяється методам соціальної інженерії, зокрема фішинговим кампаніям, які залишаються одним із найефективніших способів компрометації користувачів завдяки використанню людського фактора як найслабшої ланки. Завершальна частина розділу висвітлює інструменти аналізу шкідливого ПЗ, включаючи статичний і динамічний аналіз, використання пісочниць, поведінкової аналітики та рішень класу SIEM, SOAR, EDR і XDR. Окремо підкреслюється важливість багаторівневої стратегії захисту, яка поєднує технічні, організаційні та освітні заходи, а також роль автоматизації й моніторингу в підвищенні стійкості підприємства до сучасних кіберзагроз.

2.1. КЛАСИФІКАЦІЯ ШКІДЛИВОГО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ: ВІРУСИ, ТРОЯНИ, РУТКІТИ, БОТНЕТИ, RANSOMWARE

Шкідливе програмне забезпечення є фундаментальним елементом сучасних кіберзагроз і виступає одним із головних інструментів кіберзлочинців для компрометації інформаційних систем, викрадення даних та порушення їхньої працездатності. У наукових і практичних підходах воно розглядається як цілий спектр загроз, що відрізняються механізмами проникнення, способами поширення та рівнем впливу на цілісність, конфіденційність і доступність даних.

Шкідливе програмне забезпечення еволюціонує швидкими темпами, адаптуючись до вдосконалених механізмів захисту та експлуатуючи нові вектори атаки, зокрема у хмарних середовищах та Інтернеті речей. Воно використовує широкий набір технік розповсюдження — від традиційних заражених файлів і вкладень у листи до експлоїтів нульового дня, компрометації ланцюгів постачання та масових фішингових кампаній. Багато сучасних зразків поєднують кілька функціональних модулів: механізми проникнення, інструменти для ескалації привілеїв, канали ексфільтрації даних і компоненти, що забезпечують стійке приховане перебування в системі. Особливу загрозу становлять варіанти з підтримкою поліморфізму та руткіт-інсталяціями, які ускладнюють статичний аналіз і дозволяють уникати виявлення традиційними підходами на основі сигнатур. Інфікування IoT-пристроїв та недбалість у налаштуванні мережі створюють масштабні поверхні атаки, які зловмисники перетворюють на ботнети для DDoS-кампаній або розгортання прихованого майнінгу. Ransomware демонструє, як поєднання криптографії й бізнес-орієнтованих вимог може перетворити технічну атаку на важкий фінансовий удар для підприємств і державних установ. Водночас

зростає роль людського фактора: соціальна інженерія й цілеспрямовані spear-phishing операції залишаються ефективним способом обхідного проникнення навіть у добре захищені інфраструктури. Виявлення та реагування вимагають багаторівневих підходів — від налаштованої телеметрії та поведінкової аналітики до автоматизованих сценаріїв реагування і відновлення. Тільки комплексна стратегія, що поєднує технічні засоби, процеси управління ризиком і постійне навчання персоналу, здатна суттєво знизити ймовірність успішної компрометації та мінімізувати наслідки інцидентів.

Комп'ютерні віруси

Однією з найбільш відомих категорій є комп'ютерні віруси. Вони здатні до саморозповсюдження через зараження інших файлів, секторів пам'яті або завантажувальних областей операційної системи. Віруси часто використовують методи шифрування коду, поліморфізм і метаморфізм, що дозволяє їм обходити антивірусний захист. Хоча деякі з них можуть спричиняти лише незначні збої, інші здатні призвести до руйнування даних чи блокування критичних процесів у корпоративних мережах.

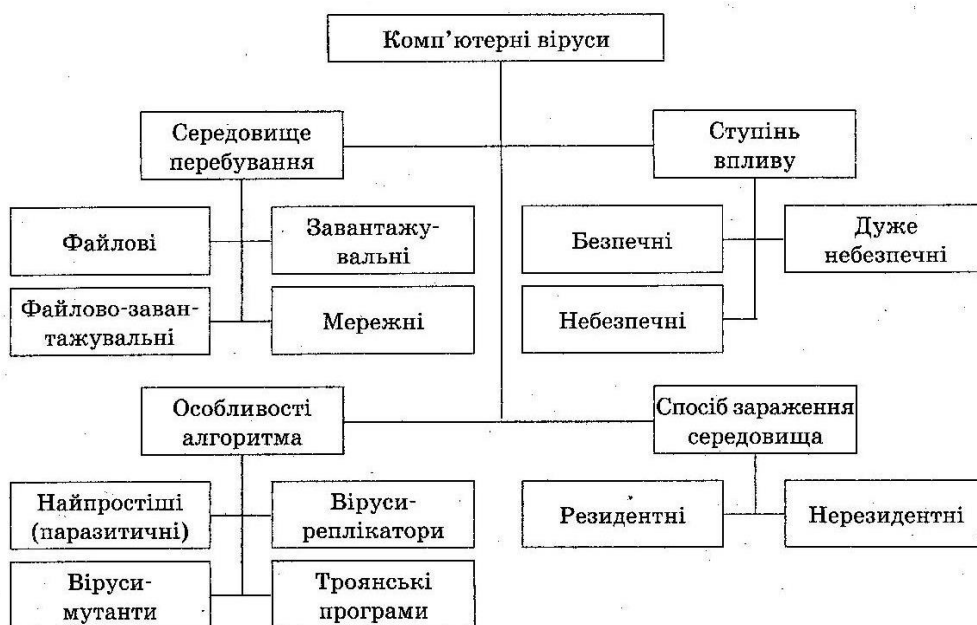


Рис. 2.1. Класифікація комп'ютерних вірусів

Джерело: [1-20, 22-29, 31-34, 45-54]

Комп'ютерні віруси класифікують за різними ознаками, що дозволяє краще зрозуміти їхню природу, механізми поширення та рівень загрози. За способом зараження виділяють файлові віруси, які вбудовуються у виконуваний файл та активуються під час його запуску, завантажувальні, що уражають сектори носіїв і запускаються разом з операційною системою, макровіруси, які використовують

уразливості офісних додатків, та сценарні, що поширюються через шкідливі скрипти у веббраузерах і поштових клієнтах. За методом поширення розрізняють резидентні віруси, які після активації закріплюються в оперативній пам'яті й здатні контролювати виклики системних функцій, та нерезидентні, що діють лише під час запуску зараженого файлу й не зберігаються в пам'яті. Важливою є класифікація за дією на систему: небезпечні віруси можуть призвести до знищення даних або порушення роботи ОС, тоді як нешкідливі обмежуються візуальними ефектами чи жартами, а шпигунські спрямовані на приховане збирання конфіденційної інформації. За структурними особливостями виділяють поліморфні віруси, здатні змінювати власний код для обходу сигнатурного виявлення, метаморфні, які повністю переписують код під час кожного нового зараження, та стелс-віруси, що маскують свою присутність у системі, підмінюючи результати перевірок.

Окремо розглядають класифікацію за середовищем поширення: мережеві віруси активно розповсюджуються через Інтернет і локальні мережі, мобільні націлені на смартфони та планшети, а хмарні орієнтовані на ураження корпоративних SaaS-сервісів і хмарних інфраструктур. Такий підхід до класифікації відображає різноманітність стратегій і методів, які застосовують зловмисники, та дозволяє формувати більш ефективні системи захисту, що враховують як класичні, так і новітні загрози.

Таблиця 2.1. Класифікація комп'ютерних вірусів: тип, характеристика, приклад та методи захисту

Тип вірусу	Характеристика	Приклад	Методи захисту
Файлові	Вбудовуються у виконуваний файли (.exe, .com), активуються під час запуску програм.	СІН (Chernobyl), Cascade	Антивірусне ПЗ, контроль цілісності файлів, обмеження прав запуску програм.
Завантажувальні	Уражають завантажувальні сектори носіїв та MBR, запускаються з ОС.	Michelangelo, Stoned	Використання UEFI/BIOS-захисту, перевірка носіїв, регулярне резервне копіювання.
Макровіруси	Використовують макромови в офісних додатках, заражають документи.	Melissa, Concept	Вимкнення макросів за замовчуванням, антивірусний моніторинг вкладень у листах.
Сценарні	Поширюються через шкідливі скрипти (VBScript, JavaScript).	LoveLetter (ILOVEYOU)	Захист поштових клієнтів, блокування скриптів, фільтрація вмісту вебсайтів.

Тип вірусу	Характеристика	Приклад	Методи захисту
Резидентні	Закріплюються в оперативній пам'яті та перехоплюють виклики системних функцій.	Randex, FuManchu	Використання IDS/IPS, системи контролю процесів, постійне оновлення ОС.
Нерезидентні	Активуються лише під час запуску зараженого файлу, не зберігаються в пам'яті.	Vienna, Sasser	Сканування файлів при запуску, контроль завантажуваних програм.
Поліморфні	Змінюють власний код при кожному зараженні, уникаючи сигнатурного виявлення.	Storm Worm, Marburg	Антивірус з евристикою, поведінковий аналіз, машинне навчання.
Метаморфні	Повністю переписують свій код, не зберігаючи початкової структури.	ZMist, Simile	Комбіновані системи виявлення, багаторівневий моніторинг, поведінковий аналіз.
Стелс-віруси	Маскують свою присутність, змінюючи відповіді системи на перевірки.	Brain, Whale	Інтегровані антивірусні рішення, контроль доступу до ядра ОС.
Мережеві	Використовують протоколи й уразливості для поширення через мережі.	Nimda, Code Red	Мережеві екрани, IDS/IPS, сегментація мереж, швидке закриття уразливостей.
Мобільні	Націлені на смартфони, поширюються через додатки та SMS.	Cabir, DroidKungFu	Захист мобільних ОС, перевірка застосунків, використання офіційних маркетплейсів.
Хмарні	Атакують SaaS-сервіси та корпоративні хмарні інфраструктури.	Cloud Snooper, Man in Cloud	Хмарні антивірусні сервіси, MFA, шифрування даних, аудит доступів.

«Структура» комп'ютерного вірусу

Під «будовою» або «структурою» комп'ютерного вірусу зазвичай розуміють набір функціональних компонентів і взаємозв'язків між ними, які забезпечують зараження, розповсюдження, приховану присутність і виконання шкідливого навантаження. Зазвичай виділяють щонайменше такі логічні блоки: механізм інфікування (інжекція в файл чи носій, експлоїт-ланцюги або дроппер/завантажувач), рушій реплікації (код, що відповідає за копіювання й розміщення свого образу в нових об'єктах), модуль персистентності (принципи забезпечення виживання після перезавантаження), модуль активації/тригерів (умови

або таймери для запуску payload), власне payload — функціонал, який завдає шкоди або виконує шпигунські/фінансові операції, і компоненти ухилення від виявлення (обфускація, шифрування, анти-детекція).

До складу складніших зразків входять конфігураційні та оновлювальні підсистеми, мережеві модулі для зв'язку з командно-контрольними серверами або для ексфільтрації даних, а також механізми самозахисту й «чистки» слідів. Архітектура може бути модульною: легкий первинний завантажувач «доставляє» основні компоненти, які потім динамічно підвантажуються — це знижує розмір початкового зразка й підвищує гнучкість.

Схеми поліморфізму та метаморфізму змінюють зовнішній вигляд коду при копіюванні, щоб уникнути сигнатурного виявлення, тоді як fileless-підходи працюють у пам'яті й майже не залишають слідів на диску. Для мережевих вірусів типовими є також DNS/DGA-модулі, проксі-шари та зашифровані канали зв'язку з інфраструктурою атакуючих.

Розуміння цих компонентів важливе не для створення шкідливого ПЗ, а щоб проектувати засоби захисту: контроль цілісності, аналіз поведінки, моніторинг пам'яті й мережі, sandboxing, ретельний менеджмент оновлень і політики доступу знижують ефективність кожного з блоків вірусу.

У виробничих середовищах найкращі практики включають багаторівневу безпеку, регулярний аудит і тестування відновлення, що робить експлуатацію вірусів економічно і технічно складнішою для зловмисників.

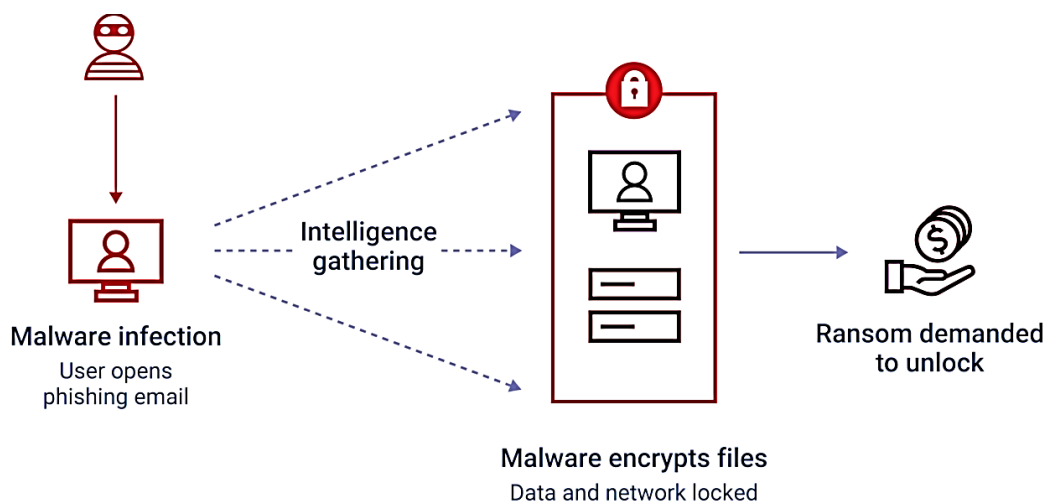


Рис. 2.2. Схеми процесу інфікування та шифрування
Джерело: [1-20]

Механізми інфікування

Механізми інфікування — це сукупність векторів і технік, через які шкідливий код потрапляє в систему або мережу жертви; вони охоплюють як класичні способи

(шкідливі вкладення у пошті, підроблені інсталятори, заражені зовнішні носії, drive-by downloads), так і сучасні канали (експлойти нульового дня, компрометація ланцюгів постачання, фішингові кампанії зі соціальною інженерією, brute-force/compromised credentials для RDP/SSH, використання вразливостей у веб-сервісах і контейнери з підміненою образою). Часто напад комбінує кілька векторів: початкове проникнення через фішинг або вразливий сервіс, потім — ескалація привілеїв і латеральний рух всередині мережі за допомогою пасивного збирання облікових даних та використання легітимних інструментів адміністрування (Living Off The Land).

Значна роль відводиться також автоматизованим інструментам — експлойт-кітам, дропперам і завантажувачам, які мінімізують початкову «вагу» зразка й дозволяють підвантажувати основні модулі вже всередині компрометованого середовища. Виявлення векторів інфікування спирається на комбінований аналіз: фільтрацію пошти і URL, sandboxing вкладень, моніторинг підозрілих інсталяторів і аномалій мережевого трафіку, а також активний threat hunting для виявлення нетипових патернів входу або незвичних процесів.

Модулі персистентності

Після успішного проникнення більшість шкідливих програм прагне закріпитися в системі, щоб пережити перезавантаження й повторні спроби видалення; для цього використовуються спеціальні модулі персистентності з широким набором технік. До них належать запис у автозавантаження (Startup folders, Run/RunOnce у реєстрі Windows), створення служб і системних таймерів (systemd unit, scheduled tasks/cron), інсталяція драйверів або kernel-модулів, модифікація завантажувача (MBR/UEFI bootkits) та прошивок пристроїв, а також використання WMI-сценаріїв, COM-об'єктів і підміни бібліотек (DLL hijacking). Сучасні загрози доповнюють ці методи хитрими прийомами: приховані/неназвані сервіси, підміна підписаних драйверів, fileless-персистентність через реєстрові або пам'яттєві дозавантаження, а також періодичні «beacon»-маячки, що запускають інжекції модулів за командою C2. Для виявлення персистентних артефактів застосовують аудит автозавантажень, перевірку підписів драйверів і бінарів, аналіз завдань і сервісів, сканування UEFI/BIOS-прошивок та пам'яті (memory forensics) — а також кореляцію цих знахідок з мережевими маячками і IOC. Ефективний захист поєднує профілактику (жорсткі політики привілеїв, application-whitelisting, Secure Boot, захищене оновлення прошивок, MFA, обмеження віддаленого доступу) із проактивним моніторингом і можливістю швидкого «containment» та відновлення із довірених резервних копій.

Узагальнюючи: механізми інфікування дають початковий доступ, а модулі персистентності — роблять цей доступ стійким; тому заходи захисту повинні одночасно закривати вектори входу й мінімізувати можливі шляхи закріплення, поєднуючи технічні контрзаходи, процеси і навчання персоналу.

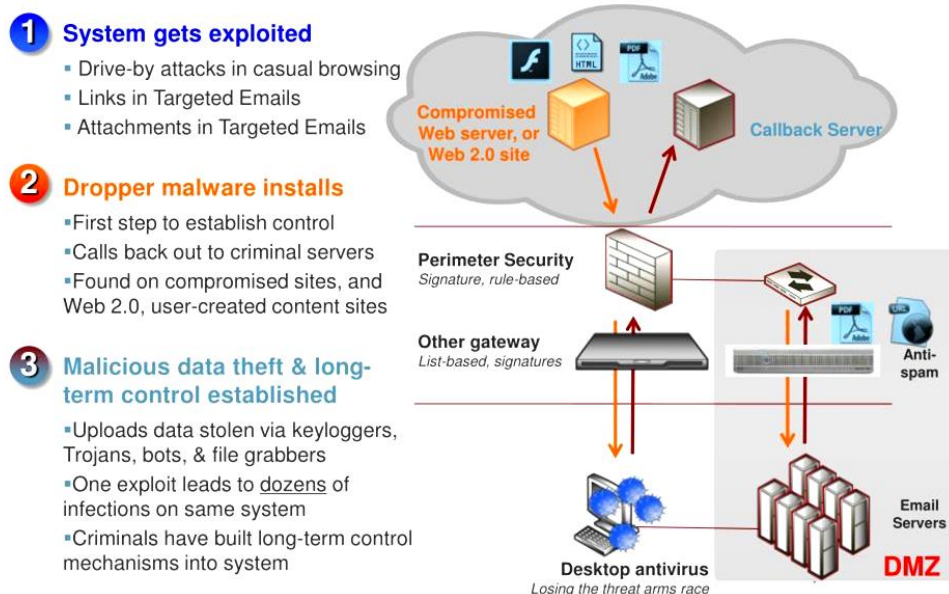


Рис. 2.3. Життєвий цикл шкідливого програмного забезпечення: експлуатація
→ дроппер → встановлення контролю
Джерело: [1-20, 26-35, 41-59]

Сучасні віруси нерідко поєднують у собі функції шпигунського ПЗ, збираючи конфіденційну інформацію користувача без його відома. Вони можуть бути запрограмовані на активацію у певний час або після виконання конкретних дій, що ускладнює їхнє виявлення. Деякі модифікації вірусів здатні використовувати вразливості програмного забезпечення для автоматичного поширення без взаємодії з користувачем. У корпоративних середовищах віруси особливо небезпечні, оскільки можуть швидко вражати мережеві ресурси та сервери, паралізуючи роботу підприємства. Вони поширюються не лише через електронну пошту чи зовнішні носії, а й за допомогою мережевих протоколів та заражених вебсайтів. Нові покоління вірусів використовують стелс-технології для маскування своєї присутності в системі. У багатьох випадках вони створюють бекдори, що відкривають шлях для інших типів шкідливого програмного забезпечення. Поява хмарних сервісів і мобільних платформ сприяла розвитку вірусів, орієнтованих на Android, iOS та SaaS-інфраструктуру. Таким чином, віруси залишаються не лише технічною, а й економічною загрозою, адже здатні завдати значних фінансових збитків. Їхнє дослідження та класифікація дозволяють створювати більш ефективні механізми захисту і прогнозувати нові тенденції у сфері кібербезпеки.

План реагування на інцидент зі шкідливим програмним забезпеченням: злагоджені дії від виявлення до відновлення

Ефективний план реагування на інцидент зі шкідливим програмним забезпеченням (malware) базується на попередній підготовці, чітко розподілених

ролях і стандартизованих процедурах, що дозволяють швидко виявляти, ізолювати, досліджувати та відновлюватися після компрометації. Підготовчі заходи включають визначення IR-лідера, SOC/triage, Forensics/Malware Analyst, IT/Ops, Network/NOC, Legal/Compliance, PR, а також контакти зовнішніх постачальників та правоохоронців; при цьому потрібно мати тестовані резервні копії (offline/immutable), налаштовані EDR/AV, NDR/IDS і SIEM, а також playbook-и й процедури chain-of-custody.

Перший сигнал (детекція) може надійти з EDR/AV, SIEM-кореляції, NDR/IDS, антивірусних сканувань або від користувача. SOC виконує первинну триаж-оцінку: підтверджує наявність індикаторів (ransom-note, незвичні процеси, підозрілі зовнішні з'єднання, нестандартні автозапуски), фіксує обсяг і важливість уражених систем і повідомляє IR-лідера. На цьому етапі важливо швидко визначити, чи йдеться про ізольований випадок чи про широку кампанію, і оцінити ризики для критичних сервісів.

Containment (ізоляція) має бути оперативним і зваженим: уражені хости ізолюють від мережі (EDR-isolate або фізичне відключення), тимчасово блокують зовнішні адміністративні канали (RDP/SSH, VPN), оновлюють firewall/DNS-правила для блокування виявлених C2-доменів або IP та за потреби сегментують мережу, щоб запобігти латеральному руху. Важливо уникати дій, які можуть стерти форензичні артефакти (не перезавантажувати критичні вузли до зняття дампу пам'яті), і координувати containment із бізнес-власниками, аби не пошкодити критичні сервіси.

Паралельно починається збереження доказів: збір знімків оперативної пам'яті, створення форензичних образів дисків (bit-stream), збереження логів (Windows Event, Sysmon, application, VPN, proxy), а також PCAP/мережових логів. Усі артефакти повинні документуватися в chain-of-custody з часовими мітками і відповідальними особами. Форензик-аналітики виконують memory-forensics (Volatility/Rekall), статичний і динамічний аналіз зразків, пошук інлайн-хуків, непідписаних драйверів, підозрілих реєстрових записів і способів персистентності.

Аналіз (Investigation) спрямований на встановлення вектора проникнення, методів розповсюдження, масштабу компрометації і наявності ексфільтрації даних. Аналітики витягують IOC (хеші файлів, домени, mutex-и, підписані конфігурації), корелюють їх у SIEM/NDR, перевіряють, чи використано легітимні інструменти адміністрування (Living off the Land) і чи скомпрометовано облікові дані або AD. На основі висновків визначаються пріоритетні блокувальні правила і подальші дії щодо containment.

Eradication (усунення) передбачає видалення шкідливих компонентів і закриття шляхів закріплення: видалення або карантин бінарів, очищення автозапусків, видалення непідписаних/підозрілих драйверів, видалення шкідливих служб і завдань, патчинг уразливих компонентів, ротація паролів і впровадження/посилення MFA. Якщо компрометація зачіпає низькорівневі

компоненти (bootkit, прошивка) або контролери домену, може знадобитись перевишивка прошивок, ребілד контролерів або апаратна заміна — рішення приймаються на підставі форензики у взаємодії з виробниками.

Recovery (відновлення) повинно виконуватися поетапно із використанням перевірених довірених резервних копій: спочатку відновлюються критичні бізнес-сервіси у контрольованому/ізолюваному середовищі з посиленням моніторингом; потім — залежні системи; перед поверненням у production кожен відновлену систему перевіряють на наявність залишкових артефактів. Важливі заходи: тестування цілісності бекапів, поступове відновлення сервісів, моніторинг на предмет повторних маячків і розгортання поліпшених політик безпеки (application whitelisting, сегментація, жорсткіші IAM).

Постінцидентний аналіз (Lessons Learned) включає after-action review з усіма зацікавленими сторонами, складання детального звіту з таймлайном і root-cause, оновлення IR-playbook-a, впровадження технічних і процесних змін та навчання персоналу. Доцільно додати в SIEM/EDR виявлені IOC, обмінятися threat intelligence з партнерами та CERT/CSIRT і запланувати повторні tabletop-вправи.

Організаційні й юридичні аспекти необхідно координувати з самого початку: повідомлення регуляторів і клієнтів при вимогах, юридична оцінка можливих наслідків, підготовка PR-комунікації і взаємодія з правоохоронцями. Рішення щодо можливого викупу (у випадках ransomware) приймаються на рівні керівництва з участю Legal і правоохоронців; загальна рекомендація — уникати виплат без ретельної юридичної та ризикової оцінки.

Ключові метрики та чек-лісти. Для оцінки ефективності відповіді відстежуйте Time to Detect (TTD), Time to Contain (TTC), Time to Remediate/Recover (TTR), кількість уражених хостів, наявність ексфільтрованих даних і бізнес-втрат. Швидкий чек-ліст першої години: 1) підтвердити інцидент і активувати IR-команду; 2) ізолювати уражені вузли (EDR-isolate); 3) зняти дамп пам'яті і форензичні образи дисків; 4) блокувати виявлені C2-IOC на DNS/firewall; 5) почати кореляцію IOC у SIEM; 6) повідомити керівництво та Legal.

Узгодженість технічних дій, форензична дисципліна, прозора внутрішня й зовнішня комунікація, а також постійне вдосконалення процедур — ось що робить IR-процес дієвим. Поєднання превентивних заходів (патчинг, MFA, резервні копії, whitelisting), детекції (EDR/NDR/SIEM) і відпрацьованих процедур реагування знижує ризики і скорочує час відновлення після інциденту зі шкідливим програмним забезпеченням.

Трояни

Інший поширений тип — трояни. Вони маскуються під легітимні програми, але насправді виконують приховані деструктивні функції. Трояни не поширюються самостійно, проте забезпечують зловмиснику віддалений доступ до системи, викрадають конфіденційні дані, дозволяють встановлювати додаткове шкідливе ПЗ

або перехоплювати дії користувача. Зокрема, банківські трояни є серйозною загрозою для фінансового сектору, оскільки спеціалізуються на викраденні реквізитів і здійсненні шахрайських транзакцій.

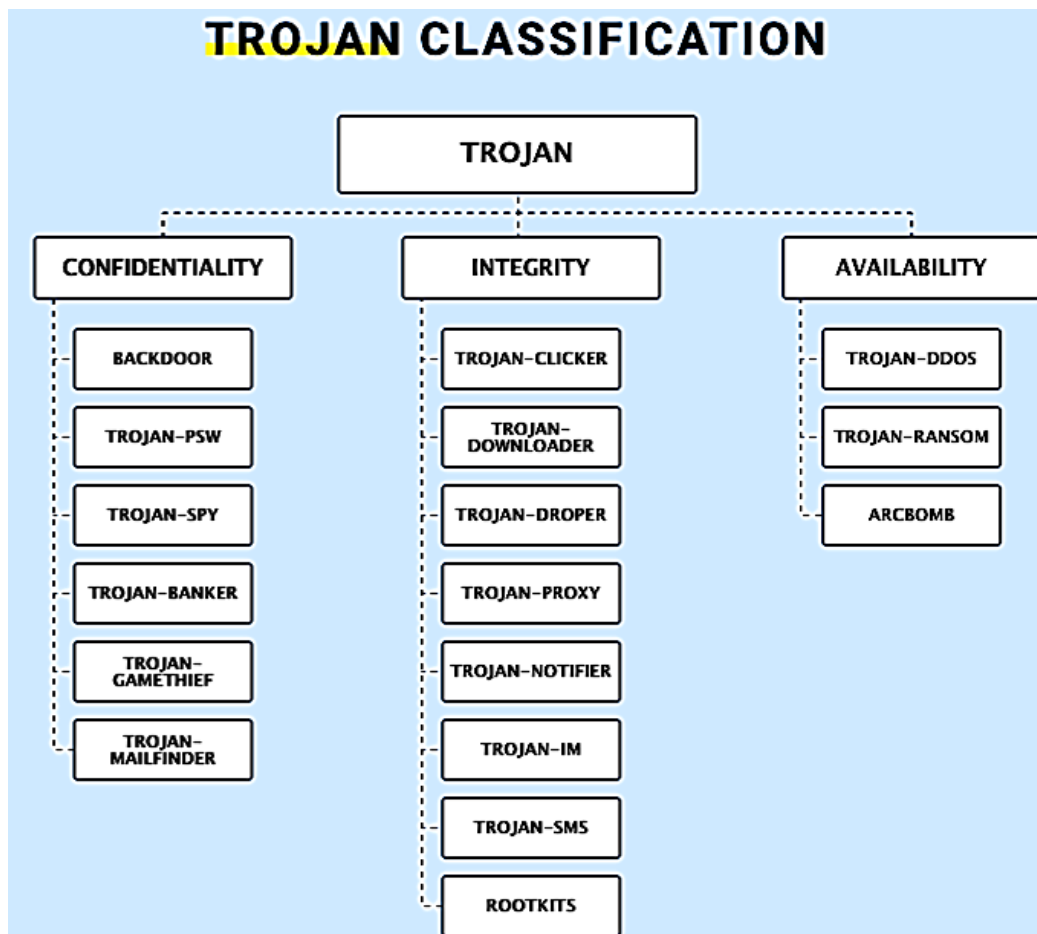


Рис. 2.4. Класифікація комп'ютерних троянів
Джерело: [1-20, 25-34, 42-56]

Трояни можуть створювати приховані бекдори, через які зловмисники отримують повний контроль над системою без відома користувача. Часто вони використовуються як початковий етап у складних багатоступеневих атаках, де відкривають шлях для інших видів шкідливого ПЗ. Завдяки своїй здатності маскуватися під легітимні програми, трояни легко обходять недосвідчених користувачів, що робить їх ефективним інструментом соціальної інженерії. Деякі сучасні зразки троянів інтегрують функції кейлогерів, модулів для зйому скріншотів і навіть шпигунських модулів для перехоплення відео та аудіо. Значна частина троянських програм поширюється через заражені вкладення електронної пошти або підроблені оновлення програмного забезпечення. В умовах корпоративних мереж трояни можуть використовуватися для збирання даних про внутрішню

інфраструктуру і підготовки до масштабних атак. Існують і так звані «Remote Access Trojans» (RAT), які забезпечують зловмиснику майже необмежений віддалений контроль над інфікованим пристроєм. У сфері кіберзлочинності трояни нерідко продаються на підпільних форумах у вигляді готових інструментів для атак. Їхня небезпека полягає також у тому, що вони залишаються непоміченими протягом тривалого часу, забезпечуючи стійку присутність у системі. Саме тому боротьба з троянами вимагає комплексного підходу, що поєднує технічні засоби захисту, регулярне оновлення програм і постійне навчання користувачів кібергігієни.

Класифікація троянських програм охоплює різноманітні підвиди, що відрізняються цілями, механізмами роботи та ступенем небезпеки для інформаційних систем, і розуміння цієї класифікації є ключовим для вибору адекватних заходів захисту. Банківські трояни орієнтовані на викрадення фінансових реквізитів і часто інтегруються у браузері або підмінюють сторінки входу, щоб перехоплювати логіни, паролі та дані карт; кейлогери фіксують кожне натискання клавіші, що робить їх ефективними інструментами для крадіжки облікових даних і конфіденційної інформації. Трояни-бекдори і Remote Access Trojans (RAT) забезпечують зловмиснику віддалений контроль над системою — запуск програм, перегляд файлів, активацію камери чи мікрофона — і часто виступають як початковий осередок для розгортання складніших атак.

Трояни-шпигуни збирають телеметрію користувача, роблять скріншоти, зчитують вміст буфера обміну та історію браузера, що використовується для промислового шпигунства або подальшої компрометації. Трояни-завантажувачі (downloaders) і дроппери служать для доставки додаткових модулів — вони мінімізують початкову «вагу» шкідника й полегшують завантаження більшої кількості компонентів після проникнення. Rogue-antivirus або фейкові захисні програми вводять користувача в оману, імітуючи сканування й вимагаючи оплати за «лікування», водночас встановлюючи бекдори або шпигунські модулі. Існують також трояни-деструктори, мета яких — порушити роботу ОС і знищити дані, що робить їх особливо небезпечними для критичних інфраструктур.

Окремою категорією можна виділити мобільні трояни, що атакують Android/iOS через шкідливі додатки або уразливі компоненти мобільних сервісів, а також трояни, орієнтовані на хмарні середовища, які прагнуть скомпрометувати облікові записи SaaS і отримати доступ до корпоративних даних. Методи поширення включають фішингові листи з шкідливими вкладеннями, підроблені оновлення ПЗ, експлойти в браузерах і уразливі плагіни, а також поставки через ланцюги постачання; унаслідок цього для захисту потрібен комплексний підхід, що поєднує контроль доступу, багатофакторну автентифікацію, моніторинг поведінки кінцевих точок, сегментацію мережі, ретельну перевірку пакунків і навчання користувачів кібергігієни.

Таблиця 2.2. Класифікація троянських програм: типи, механізми дії, приклади та методи виявлення

Тип трояна	Механізм дії	Приклад	Методи виявлення
Банківський троян (banker)	Інжекція в процеси браузера, підміна сторінок входу, перехоплення форм та Web-injection, перехоплення 2FA/OTP.	Zeus, Dridex, Emotet (як компонент)	Моніторинг HTTP/HTTPS поведінки, аналіз вставок у веб-сторінки, поведінковий аналіз браузерної активності, ML-аналіз транзакцій.
Кейлогер (keylogger)	Перехоплення натискань клавіш, API-хукінг, інжекція у ввідні підсистеми, запис буфера обміну.	DarkHotel-style modules, Beast keylogger	Аналіз процесів для перехоплення вводу, EDR-функції для виявлення API-хукінгу, моніторинг підозрілих файлів журналів/вивантажень.
Бекдор / Remote Access Trojan (RAT)	Відкриття каналу C2, віддалене виконання команд, доступ до файлів/камер/мікрофона, рух латерально по мережі.	DarkComet, Gh0st RAT, Remcos	Аналіз мережевого трафіку (C2), виявлення аномальних з'єднань, форензика пам'яті, індикатори поведінки (unexpected remote control).
Завантажувач / Downloader	Містить мінімальний код для контакту з C2 та завантаження/інсталяції основних модулів.	Різні початкові ланки Emotet/TrickBot	Виявлення доменів C2, блокування підозрілих завантажень, аналіз поведінки процесів після запуску, захист пошти.
Шпигунський троян (spy)	Збір телеметрії: скриншоти, історія браузера, файли, метадані; ексфільтрація через стеганографію або C2.	FinFisher (кілька компонентів), DarkHotel modules	DLP (Data Loss Prevention), аналіз аномалій у трафіку ексфільтрації, моніторинг файлових операцій, EDR/UEBA.
RAT-мобайл (mobile trojan)	Отримання прав на пристрої, підміна SMS/OTP, фальшиві додатки, експлуатація дозволів.	LokiBot (мобільні версії), Anubis	Моніторинг дозволів додатків, контроль встановлення з неперевіраних джерел, мобільні EDR, аналіз поведінки додатків.
Rogue/Scareware (фейкові антивіруси)	Імітація сканування, видача фальшивих результатів,	FakeAV сімейства (FakeRean,	Аналіз процесів/реєстру, виявлення підозрілих GUI-індикаторів,

Тип трояна	Механізм дії	Приклад	Методи виявлення
	вимога оплати, часто встановлюють бекдори.	Win32/Scareware)	блокування підозрілих інсталяторів, перевірка цифрових підписів.
DDoS-троян / бот-компонент	Інфекція для приєднання до ботнету, отримання команд для генерування трафіку або участі в атаках.	Mirai-подібні модулі, Kaiten	NDR/IDS для виявлення аномального вихідного трафіку, аналіз поведінки мережі, підключення до threat intelligence щодо IoC.
Троян-деструктор (wiper)	Шифрування або безповоротне стирання файлів, руйнування завантажувача/конфігурацій; може маскуватися під ransomware.	Shamoon, NotPetya (wiper-компоненти)	Моніторинг доступу до масивних операцій запису/видалення, контроль цілісності файлів, швидке виявлення аномалій у файлових операціях, резервне копіювання.
Модульні / багатофункціональні трояни	Мають модульну архітектуру: завантажувач, шпигун, banker, проксі тощо; адаптуються під завдання.	TrickBot, BazarLoader	Комбінація мережевого та поведінкового моніторингу, кореляція IOC у SIEM, форензика пам'яті/диска, сегментація мережі.

Виявлення троянів ускладнюється їхньою здатністю приховуватися, використовувати шифрування каналів зв'язку та модульну архітектуру, тому ефективні заходи захисту ґрунтуються на поєднанні сигнатурних, евристичних і поведінкових методів аналізу, регулярних оновлень і швидкому реагуванні на інциденти з відновленням із резервних копій.

Під «структурою трояна» зазвичай розуміють набір функціональних модулів і зв'язків між ними, які разом забезпечують доставку, інсталяцію, приховану присутність, виконання деструктивного або шпигунського навантаження та підтримку зв'язку з оператором. На базовому рівні типовий троян складається з завантажувача (loader/dropper), який виконує початкове розгортання — він може приходити у вигляді підробленого інсталятора, вкладення в листі або як частина експлойт-кейту, і його завдання — доставити основний модуль у середовище жертви, іноді зв'язавшись із сервером управління для завантаження додаткових компонентів. Далі слідує основний рух реплікації й персистентності: модулі, що забезпечують закріплення в системі (ключі автозавантаження, служби, планувальники завдань, інсталяція драйверів або модифікація завантажувача/прошивки), — саме вони гарантують виживання після перезавантаження й ускладнюють видалення.

Компонент зв'язку з оператором (C2) реалізує протокол зворотного зв'язку — HTTP/HTTPS-запити, P2P-мережі, DGA-домени або тор-канали — і передає команди, модуля для оновлення та шляхи ексфільтрації даних. Сам payload — функціональні модулі трояна — може бути модульним: кейлогер, скрипт для збирання файлів, інжектор у процеси, проксі/реле для латерального руху, компонент для виконання DDoS або для крадіжки банківських даних; модульність дозволяє зловмисникам завантажувати потрібні інструменти «на вимогу». Для збереження стелсу й уникнення аналізу в трояні вбудовують механізми ухилення: обфускацію коду, шифрування конфігурацій, анти-сандбокс і анти-детекторні перевірки середовища (перевірка на віртуалізацію, таймінги, наявність інструментів аналізу), поліморфні/метаморфні механізми та «fileless»-техніки, що працюють у пам'яті.

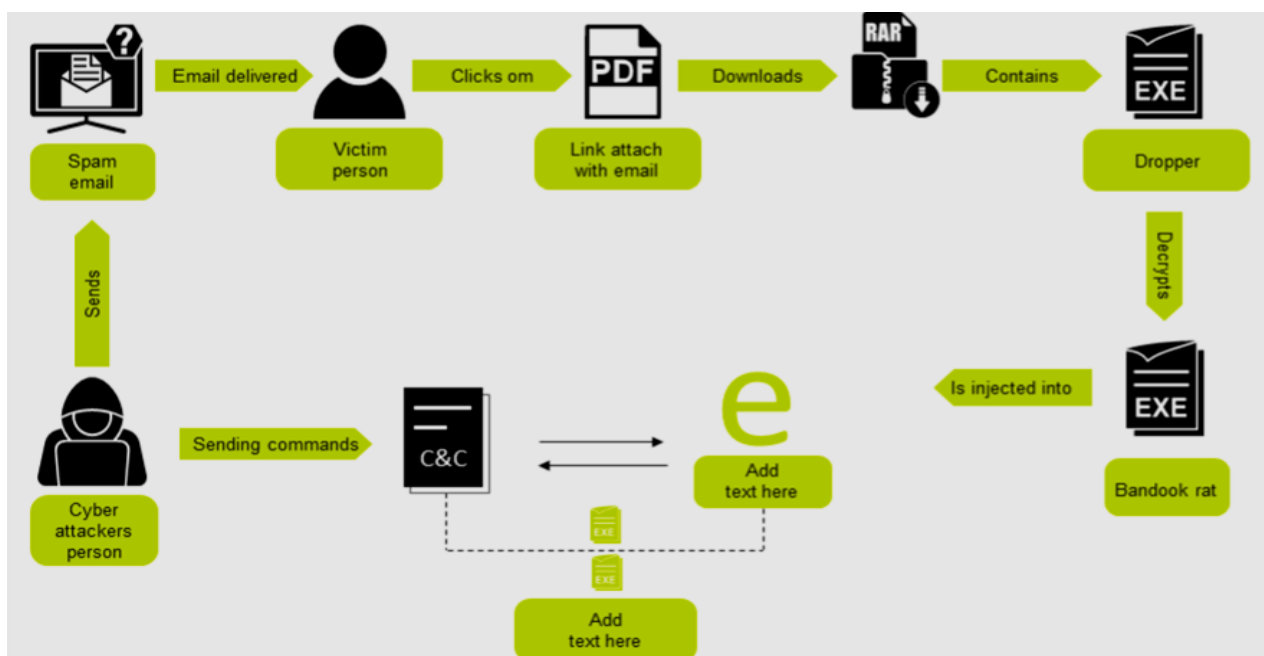


Рис. 2.5. Схема поширення трояна через фішингову розсилку: від доставки шкідливого вкладення до встановлення дроппера, розгортання бекдора (Bandoor) та встановлення зв'язку з C2-сервером
Джерело: [1-20, 28-36]

У просунутих сімействах присутні підсистеми самозахисту (виявлення інструментів безпеки, руйнування слідів логів), механізми оновлення (автономне підвантаження нових версій або плагінів), конфіг-файли з налаштуваннями C2/ключів і модулі монетизації (майнінг, проксі, продаж доступу). Архітектурно троян часто має централізовану або модульно-агентську структуру: «легкий» стартовий дроппер мінімізує розмір початкового файлу й підвантажує вагоміші компоненти після встановлення, що ускладнює статичний аналіз. Для forensic-аналізу важливими артефактами є конфігурації, рядки з'єднань C2, збережені ключі

й індикатори компрометації; для захисту проти троянів рекомендовано поєднання превентивних заходів (application whitelisting, MFA, жорстка політика прав, патчинг), виявлення на основі поведінки (EDR, SIEM, UEBA), регулярний аудит автозавантажень і моніторинг мережесих маячків, а також перевірені процедури інцидент-респонсу і відновлення із довірених резервних копій.

План реагування на інцидент із трояном: злагоджені дії від виявлення до відновлення

Ефективне реагування на інцидент, пов'язаний із троянським ПЗ, починається з підготовки: визначення ролей і каналів зв'язку (IR-лідер, SOC, IT/Ops, forensic-аналітик, мережеві операції, юридичний відділ і PR), тестування процедур збору доказів і наявності перевірених резервних копій. Необхідно заздалегідь налаштувати і підтримувати EDR/AV, SIEM і NDR, визначити критичні бізнес-сервіси та мати контакти постачальників і правоохоронних органів — це скорочує час реагування й підвищує якість прийнятих рішень у момент інциденту.

Після первинного сигналу (спрацювання EDR, аномалія в мережі, повідомлення користувача) SOC проводить швидку триаж-оцінку, щоб відфільтрувати хибні спрацьовування і підтвердити індикатори компрометації: наявність незвичних процесів, підключень до підозрілих доменів, нові автозавантаження або змінені системні бінарі. Якщо підтверджується присутність трояна або бекдора, IR-лідер активує команду, інформує керівництво і юридичний відділ, після чого пріоритетом стає containment — зупинка подальшого поширення і блокування каналів управління.

Containment виконується координовано: уражені хости ізолюються від мережі (через EDR-isolate або фізично), підозрілі облікові записи тимчасово блокуються, RDP/SSH і зовнішні адміністративні доступи обмежуються, а firewall/DNS правила оновлюються для блокування виявлених C2-доменів чи IP. У той же час необхідно уникати поспішних дій, які могли б знищити важливі докази — наприклад, перезавантаження інфікованого хоста до зняття memory-dump може стерти сліди інлайн-хуків чи завантажених модулів.

Паралельно з containment починається етап збереження доказів: робляться знімки оперативної пам'яті, створюються форензичні іміджі дисків, зберігаються логи (Windows Event, Sysmon, VPN, proxy, firewall) і мережеві захоплення (PCAP). Всі дії документуються у chain-of-custody з відмітками часу й відповідальними, що робить отримані артефакти придатними для подальшого аналізу й, за потреби, для правоохоронних органів.

Forensic-команда і malware-аналітики розгортають глибокий аналіз: статичний та динамічний розбір бінарів, memory-forensics (Volatility/Rekall) для пошуку інлайн-хуків і невластивих структур, аналіз незвичних драйверів чи змін в реєстрі, а також кореляція мережесих IOC у SIEM/NDR для визначення повного охоплення компрометації й можливих шляхів lateral movement. Важливо з'ясувати, чи троян

використовував легітимні інструменти (Living off the Land), чи викрадені облікові дані, адже це впливає на обсяг відновлювальних робіт (заміна паролів, аудит облікових записів).

Після ідентифікації векторів проникнення і методів дії слідує етап усунення: видалення шкідливих компонентів, ревізія автозавантажень, видалення підозрілих служби й запланованих завдань, видалення невідомих драйверів та блокування завантажувальних уразливостей. Якщо AD/контролери або критичні інфраструктурні елементи скомпрометовані, може знадобитися їхнє відновлення з довірених образів або rebuild; перед відновленням системи обов'язково проводиться повна перевірка резервних копій на відсутність атакуючих артефактів. Ротація облікових даних і впровадження/посилення MFA для адміністративних облікових записів є обов'язковими заходами.

Етап відновлення здійснюється поетапно: спочатку відновлюються критичні сервіси у контрольованому середовищі з активним моніторингом, потім — залежні системи; кожна відновлена система проходить перевірку на повторну активність трояна перед поверненням у продакшн. Одночасно впроваджуються додаткові захисні правила (жорсткіші політики доступу, application-whitelisting, оновлення патчів), щоб зменшити ризик повторної компрометації.

Постінцидентний аналіз — невід'ємна частина playbook: проводиться after-action review з усіма учасниками, складається докладний звіт із таймлайном подій, root-cause аналізом, переліком виявлених IOC і оцінкою збитків. На основі висновків оновлюються IR-playbook, політики, конфігурації моніторингу і проводяться навчання персоналу (фішинг-тести, вправи з реагування). Також важливо розподілити відповіді на питання взаємодії з партнерами, постачальниками і, за потреби, правоохоронними органами. Крім оновлення процедур, результати постінцидентного аналізу мають бути формалізовані у вигляді actionable recommendations — конкретних технічних і організаційних змін (патчі, нові детекції, зміни політик доступу, додаткові тренінги) з призначеними відповідальними та термінами виконання. Важливо також передати релевантну розвідку (IOC, TTP — Tactics, Techniques and Procedures) у внутрішні системи threat-intelligence і, за можливості, обмінятися нею з галузевими партнерами або CERT/CSIRT для попередження подібних атак у екосистемі. Після впровадження змін необхідно провести верифікацію — regression-тести детекцій, повторні tabletop-вправи і технічні перевірки (penetration testing, red-team) щоб переконатися в реальному підвищенні стійкості. Нарешті, результати і виконання рекомендацій треба відслідковувати через KPI і регулярні аудит-рев'ю, аби закрити виявлені прогалини і виміряти фактичне покращення Time To Detect (TTD), Time To Contain (TTC) та інших показників ефективності.

Критично важливо визначати й відстежувати метрики ефективності реагування: Time to Detect (TTD), Time to Contain (TTC), Time to Remediate/Recover

(TTR), кількість вражених хостів і наявність викрадених даних. Ці показники дозволяють вимірювати успіх заходів і планувати пріоритетні інвестиції у захист. Дані для цих метрик потрібно збирати централізовано з джерел телеметрії (EDR, NDR, SIEM, SOAR, лог-сервери) і візуалізувати у дашбордах для оперативного моніторингу й звітності. Важливо встановлювати базові рівні (baseline) і цільові значення по кожній метриці, диференціюючи по ступеню критичності активів і типу інциденту, щоб правильно пріоритизувати відповіді та інвестиції. Результати метрик використовуються в after-action review (post-incident review) для виявлення кореневих причин, оцінки ефективності playbook-ів і обґрунтування ROI (рентабельності інвестицій) у додаткові датчики або автоматизацію. Крім того, метрики слід валідувати через регулярні вправи (tabletop, red/blue/purple-team), автоматичні алерти при перевищенні порогів і періодичну ревізію KPI відповідно до еволюції загроз.

У підсумку, план реагування на інциденти з троянами поєднує технічні, організаційні й юридичні заходи: від швидкої ізоляції і збереження доказів через глибокий forensic-аналіз до системного відновлення та навчання — все це створює стійкий механізм, що мінімізує шкоду, прискорює відновлення і знижує ймовірність повторних компрометацій.

Руткіти

Руткіти відомі з ранніх 1990-х і неодноразово використовувалися як складова високотехнологічних атак проти корпорацій і державних структур (приклади: сімейства руткітів, що використовувалися в APT-кампаніях та bootkit-операціях). Аналіз історичних інцидентів підтверджує, що руткіти часто застосовуються для довготривалої розвідки, ексфільтрації даних і підготовки складних атак. Основна функція полягає не у безпосередньому нанесенні шкоди, а у приховуванні слідів іншого шкідливого ПЗ. Вони інтегруються у ядро операційної системи, змінюють драйвери й системні бібліотеки, приховуючи процеси, файли та записи реєстру. Руткіти є складними для виявлення навіть сучасними засобами захисту, що робить їх небезпечним інструментом тривалого несанкціонованого контролю над інформаційними системами.

Виявлення руткітів потребує спеціалізованих технік — порівняння образів пам'яті та диска, пошуку інлайн-хуків, непоєднаних або непідписаних драйверів і невідповідностей у внутрішніх структурах ядра. Інструменти пам'яттєвої форензики (наприклад, Volatility або Rekall) і аналіз UEFI/BIOS-прошивок (UEFI — Unified Extensible Firmware Interface) допомагають ідентифікувати завантажувальні ін'єкції та модифікації прошивки. Для профілактики і значного ускладнення можливості інсталяції руткітів застосовують апаратні механізми захисту (TPM — Trusted Platform Module, Secure Boot, Measured Boot), політики підписування драйверів і жорстке обмеження прав на встановлення низькорівневого програмного забезпечення. У разі підтвердженого ураження часто необхідне повне відновлення з

довірених образів, перевишивка прошивок або апаратна заміна, а також тісна координація з виробниками обладнання, постачальниками EDR/AV і, за потреби, з правоохоронними органами.

Схема інсталяції руткіта ілюструє послідовність дій від початкового дроппера або експлойта до повноцінного ядрового компоненту, який надає зловмиснику стійкий, прихований доступ до системи. Зазвичай процес починається з початкового проникнення — користувацький процес (вкладення в листі, шкідливий інсталятор або експлойт веб-сервісу) запускає дроппер, який у свою чергу виконує ескалацію привілеїв або експлуатує вразливість для переведення виконання в привілейований режим. Після отримання достатніх прав дроппер встановлює перший етап ядрового коду — підписаний або прикритий драйвер, bootloader-патч (bootkit) або інший модуль, який завантажується до kernel-mode і реалізує механізми персистентності. Типові ядрові техніки включають патчинг таблиці системних викликів (SSDT), модифікацію таблиці переривань (IDT), ін'єкцію/інлайн-хукінг у ядрові функції, зміну IRP/dispatch-таблиць драйверів і пряме маніпулювання внутрішніми структурами ядра (DKOM — Direct Kernel Object Manipulation). Інший розповсюджений підхід — інсталяція filter-/minifilter-драйверів (наприклад файлові або мережеві фільтри), які працюють легітимно на kernel-рівні, але виконують приховані операції (перехоплення файлових операцій, приховування процесів, перехоплення мережевого трафіку).

Щоб залишатися невидимим, ядровий компонент зазвичай реалізує кілька анти-форензичних прийомів: підміняє відповіді на виклики діагностичних інструментів, видаляє сліди з таблиць процесів і файлових систем, шифрує свої конфігурації і використовує періодичні «маячки» (beaconing) для зв'язку з C2. На сучасних ОС (наприклад Windows x64) частину простих патчів SSDT перекриває PatchGuard і механізми підписування драйверів, тому складніші руткіти обходять обмеження через підміни прошивок/завантажувача (bootkit/UEFI) або через заглиблення в механізми віртуалізації (гіпервізорні руткіти). Для виявлення та аналізу таких ядрових руткітів аналітики застосовують поєднання пам'яттєвої форензики (memory forensics) і порівняння образів диска та оперативної пам'яті — це дозволяє виявити невідповідності між тим, що збережено на носії, і тим, що реально виконується в системі.

Апаратні механізми захисту й верифікації завантаження, такі як TPM (Trusted Platform Module) і Measured Boot, а також Secure Boot і Kernel-mode Code Signing / HVCI (Hypervisor-protected Code Integrity), значно ускладнюють можливість успішної інсталяції непідписаних або модифікованих драйверів. У разі підозри на компрометацію прошивки або UEFI (Unified Extensible Firmware Interface) часто єдиним гарантійним шляхом усунення є перевишивка (reflash) або заміна пристрою, а відновлення ОС має виконуватися з довірених, перевірених образів; просте видалення модулів із системи зазвичай не гарантує повного усунення загрози. Крім

технічних заходів, критично важлива організаційна готовність: затверджені playbook-и для incident response (реагування на інциденти), швидка координація з постачальниками ПЗ/обладнання і, при необхідності, залучення виробника або правоохоронних органів для коректного розв'язання інциденту.

Виявлення і аналіз таких інфекцій потребують спеціалізованих технік: порівняння образів ядра на диску і в пам'яті, пошук інлайн-хуків, аналіз незвичних або непідписаних драйверів, memory-forensics інструменти (наприклад Volatility/Rekall), а також аналіз логів «маячків» у мережевому трафіку. Практичні заходи захисту включають запобіжні політики (заборона встановлення непідписаних драйверів, застосування Secure Boot і HVCI/Kernel-mode code signing), жорстку адміністративну сегментацію, регулярний патчинг, моніторинг підписів драйверів і цілісності завантажувача, а також захищене оновлення прошивок. У разі виявлення інфікування часто потрібне відновлення з довірених резервних копій, пере-прошивка апаратних компонентів або повне перевстановлення ОС; при ураженні прошивки/UEFI може знадобитися заміна або перевишивлення пристрою. Такий захисний підхід (профілактика, моніторинг, форензика і чіткі процедури реагування) істотно ускладнює можливість успішної інсталяції й довготривалого прихованого функціонування руткіта.

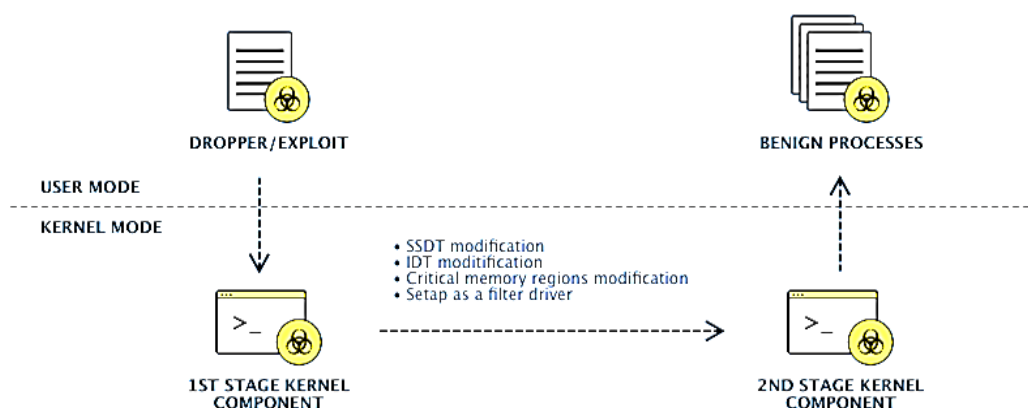


Рис. 2.6. Схема інсталяції руткіта: перехід від дроппера/експлойта до kernel-mode компонентів (приклади: SSDT/IDT-патчинг, критичні модифікації пам'яті, filter-драйвери)

Джерело: [1-20, 23-38]

Руткіти забезпечують стійку присутність у системі через механізми персистентності — модифікацію завантажувальних процесів, інтерцепцію системних викликів і підміну API, що дозволяє їм зберігати контроль після перезавантаження. Існують як user-mode, так і kernel-mode руткіти, причому останні більш небезпечні через доступ до привілейованих ресурсів і можливість модифікації ядра; окремо виокремлюють bootkit-варіанти, що уражають прошивку чи завантажувач (UEFI/MBR). Через глибоку інтеграцію у систему руткіти часто

уникають виявлення традиційними сканерами й можуть блокувати або підмінювати результати інструментів безпеки, ускладнюючи нормальні процедури інцидент-пону.

Виявлення руткітів потребує спеціалізованих підходів — контроль цілісності на рівні прошивки й ядра, аналіз незвичних патернів поведінки, порівняння образу пам'яті з довіреними еталонами та використання апаратно-орієнтованих засобів моніторингу. Лікування інфікованих систем часто передбачає повне відновлення з апаратно-чистих резервних копій або перевстановлення ОС разом із прошивкою пристроїв, оскільки просте видалення компонентів руткіта зазвичай не гарантує повного усунення загрози. Запобігання поширенню руткітів має базуватися на багаторівневих заходах: політиках мінімальних привілеїв, захищених механізмах завантаження (Secure Boot), регулярних оновленнях прошивок та посиленому контролю ланцюгів постачання ПЗ.

Класифікація руткітів охоплює кілька взаємопов'язаних категорій, які відрізняються рівнем інтеграції в систему, механізмами приховування та наслідками для безпеки; розуміння цих розрізень є ключовим для вибору адекватних підходів до виявлення та нейтралізації. Користувацькі руткіти працюють у просторі користувача, змінюючи бібліотеки й утиліти, підмінюючи інструменти діагностики та маскуючи процеси, файли чи мережеві з'єднання, тому їхню присутність іноді можна помітити через аномалії в поведінці додатків або невідповідності у контрольних сумах. Руткіти ядра інтегруються в саме ядро операційної системи або встановлюють шкідливі драйвери, перехоплюючи системні виклики і отримуючи привілеї, які дозволяють їм залишатися практично невидимими для стандартних антивірусних засобів. Bootkit-варіанти уражають етапи завантаження — MBR або UEFI — і активуються до запуску ОС, що дає їм перевагу в приховуванні і робить їх стійкими до простих перевстановлень системи. Прошивкові руткіти вражають мікропрограми пристроїв (прошивки дисків, мережевих карт, контролерів), завдяки чому можуть переживати форматування й відновлення ОС, створюючи довготривалу загрозу на апаратному рівні. Віртуалізаційні чи гіпервізорні руткіти працюють ще глибше — підмішуючи або контролюючи гіпервізор, вони можуть маніпулювати робочими системами так, щоб жертва не підозрювала про компрометацію.

Кожен із типів руткітів вимагає специфічних методів виявлення: від аналізу цілісності образів і контрольних сум на рівні ядра та прошивки до поведінкової телеметрії, порівняння з «чистими» еталонами та застосування апаратних засобів контролю (TPM, Secure Boot). Лікування інфікованих систем часто передбачає не лише видалення шкідливих модулів, а повну ревізію і, за потреби, заміну або перевишивлення апаратних компонентів і відновлення з довірених резервних копій. Запобігання поширенню руткітів має базуватися на багаторівневому захисті: політиках мінімальних привілеїв, захищених механізмах завантаження, регулярних

оновленнях прошивок, ретельній перевірці ланцюгів постачання ПЗ і постійному моніторингу аномалій у поведінці системи. Такий підхід знижує ймовірність успішної інсталяції руткіта і підвищує шанси на своєчасне виявлення та ефективне реагування.

Таблиця 2.3. Класифікація руткітів: типи, механізми дії, приклади та методи виявлення

Тип руткіта	Механізм дії	Приклад	Методи виявлення
User-mode rootkit (користувачього рівня)	Підміна/патчинг бібліотек і користувацьких утиліт, інтерцепція бібліотечних викликів (API hooking), приховування файлів/процесів у просторі користувача.	Зламані версії утиліт, патчені DLL	Порівняння контрольних сум і підписів файлів, моніторинг незвичних процесів/мережевої активності, сканування з Live-CD, аналіз поведінки EDR.
Kernel-mode rootkit (ядровий руткіт)	Інсталяція шкідливих драйверів або модифікація таблиць системних викликів (SSDT), перехоплення IRQ/ драйверів — повний контроль над ОС.	Руткіти, що патчать ntoskrnl або встановлюють драйвер з підписом	Порівняння образу ядра з еталоном, аналіз незвичних поведінкових патернів, інструменти пам'яті (Volatility), виявлення незвичних/несигнатурних драйверів, контроль підписів драйверів.
Bootkit / MBR / UEFI rootkit	Ураження етапів завантаження (MBR, VBR) або прошивки UEFI — виконується до ОС, забезпечує персистентність і прихованість.	Bootkit, що модифікує MBR або інфікує UEFI	Верифікація Secure Boot/Measured Boot логів, аналіз прошивки, сканування носіїв з зовнішнього середовища, порівняння образів завантажувача, апаратна телеметрія (TPM).
Firmware rootkit (прошивковий)	Інфікування мікропрограм пристроїв (HDD, NIC, контролери), що дозволяє зберігатися після перевстановлення ОС.	Rootkits у прошивці дисків або мережевих карт	Аналіз та порівняння прошивок, інструменти верифікації прошивки виробника, апаратні журнали, поведінковий моніторинг пристроїв, апаратна ізоляція.
Hypervisor / VMM rootkit (гіпервізорний)	Підміна або інсталяція гіпервізора під ОС (або рівня VMM), перехоплення всіх	Blue Pill (гіпотетичний), підміна гіпервізора	Вимірювання часу виконання (timing attacks), VM-introspection, апаратні показники віртуалізації, аналіз несумісностей

Тип руткіта	Механізм дії	Приклад	Методи виявлення
	системних викликів ОС.		CPUID/timing, зовнішній моніторинг хосту.
Fileless / memory-resident rootkit	Робота виключно в пам'яті без постійних файлів на диску; використовує легітимні інструменти (Living off the Land).	Скриптові інжекції в пам'ять процесів	Пам'ятева форензика, аналіз дзеркал пам'яті (RAM dump), поведінковий моніторинг процесів, EDR із захопленням стеків/інжекцій.
Network / router / IoT rootkit	Інфікування прошивок мережевих пристроїв або вбудованих систем для перехоплення трафіку і персистентного доступу.	Руткіти в прошивці роутера, Mirai-подібні модулі	Аудит конфігурацій, аналіз мережевого трафіку на аномалії, перевірка контрольних сум прошивок, відновлення з довірених образів, сегментація мережі.
Hybrid / modular rootkit	Модульна архітектура: комбінація user/kernel/firmware компонентів з віддаленим завантаженням модулів.	Складні APT-руткіти з кількома компонентами	Комплексний підхід: мережевий моніторинг, EDR/EDR+NDR, форензика пам'яті/диска, інтеграція з SIEM, кореляція IOC.

Для виявлення та розслідування інфікування руткітами аналітики використовують набір спеціалізованих інструментів як для пам'яттевої форензики, так і для аналізу на рівні прошивок і ядра. Серед ключових інструментів — Volatility / Rekall (memory forensics — аналіз знімків оперативної пам'яті, пошук інлайн-хуків, невідповідностей у структурах процесів і незвичних драйверів), UEFITool та Chipsec (аналіз UEFI/BIOS-прошивок і перевірка цілісності мікрокоду), Binwalk і спеціальні firmware-розбірники для витягання і дослідження образів прошивок. Для порівняння образів ядра і виявлення модифікацій використовують diff-інструменти та побайтний аналіз образів, а також утиліти для перевірки цифрових підписів і контрольних сум. Динамічний аналіз і трасування реалізують через sandbox-середовища (наприклад, Cuckoo Sandbox) та інструменти трасування подій ОС (ETW/strace/dtrace), щоб відстежити завантаження драйверів, виклики API та незвичну поведінку. Мережевий аспект досліджується за допомогою NDR/IDS (наприклад Suricata/Zeek) для виявлення beacon-патернів і аномальних з'єднань до потенційних C2-серверів.

Сучасні руткіти все частіше націлені на ранні етапи завантаження та на прошивки пристроїв, тому апаратно-орієнтований захист має вирішальне значення. Технології на кшталт TPM і Measured Boot дозволяють відстежувати цілісність

завантажувальних компонентів, тоді як Secure Boot і Kernel-mode Code Signing / HVCI (Hypervisor-based Code Integrity) обмежують можливість виконання непідписаного чи небезпечного коду в привілейованому режимі. Наявність цих механізмів суттєво знижує ймовірність успішної ін'єкції ядрових компонентів, бо атакувальнику доводиться обходити апаратні виміри або підміняти прошивку — складні й ризиковані операції, які залишають більше артефактів для виявлення. Однак захист на апаратному рівні ефективний лише за умови правильної конфігурації й оновлення — наприклад, Secure Boot та перевірка підписів драйверів мають доповнюватися управлінням ключами та процесом довірених оновлень прошивок.

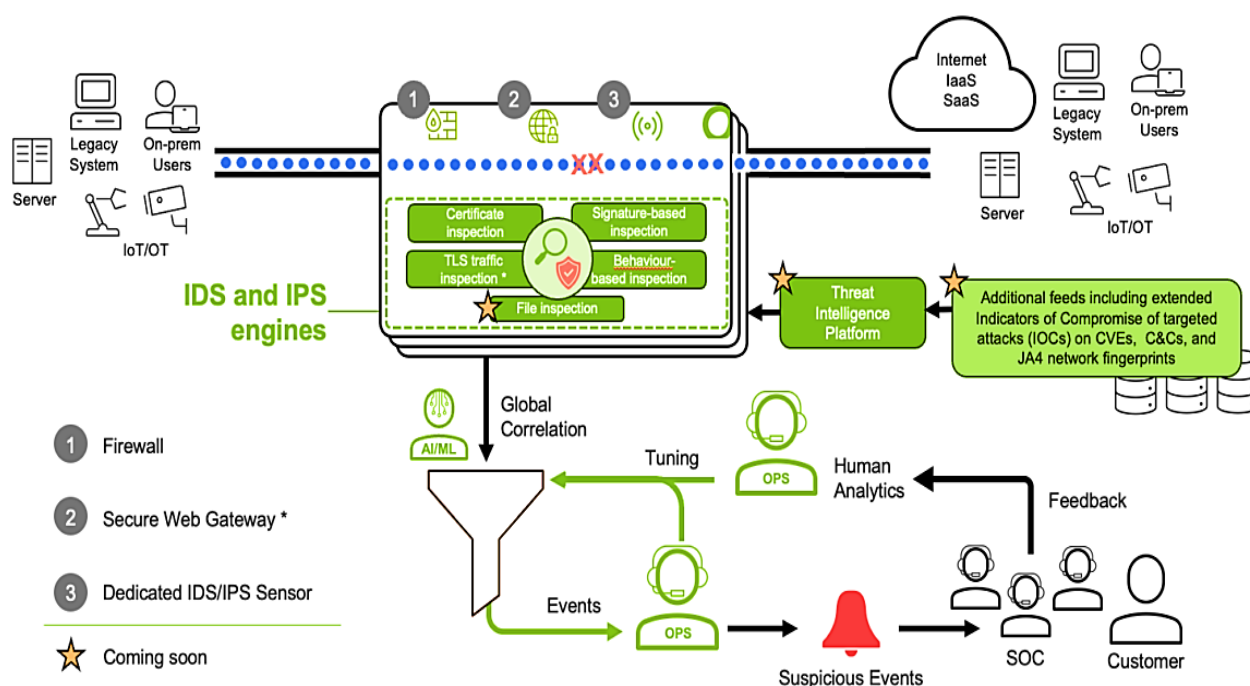


Рис. 2.7. Архітектура мережевого виявлення і кореляції подій: IDS/IPS → NDR → кореляція з Threat Intelligence і процес реагування (SOC / SOAR)
Джерело: [1-20, 23-38]

Схема ілюструє типову архітектуру виявлення мережевих загроз: перетин TLS/HTTP/файлів на межі, інспекція на рівні підписів і поведінки, кореляція подій з платформою threat-intelligence, та подальше опрацювання інцидентів SOC/OPS з використанням автоматизованих playbook.

Рекомендовано поєднувати технічні контрзаходи: контроль цілісності (файлової системи, boot-секторів і прошивок), EDR із моніторингом пам'яті (щоб ловити fileless-ін'єкції й інлайн-хуки), регулярні сканування на предмет непідписаних або аномальних драйверів, та NDR/IDS для виявлення регулярних беасоп-патернів і незвичного вихідного трафіку. Важливий елемент — періодичні

офлайн-перевірки образів систем (порівняння образу диска і образу пам'яті з еталоном), а також тестування відновлення з резервних копій після імітованих інцидентів. Автоматизовані playbook-и (SOAR) для containment-дій — ізольоване блокування підозрілих вузлів, збір знімків пам'яті і логів, та кореляція з IOC — значно пришвидшують реагування.

Технічних інструментів недостатньо без чіткої організаційної бази: необхідні політики мінімальних привілеїв, контроль доступу до адміністративних інтерфейсів (RDP/SSH) з обмеженням і MFA, процеси управління змінами прошивок і драйверів, а також обов'язковий аудит ланцюга постачання ПЗ (SBOM — software bill of materials) для зниження ризику compromise через залежності. Важлива співпраця з постачальниками обладнання: вимоги до захищених оновлень, підписування прошивок і оперативний обмін інформацією про вразливості. Також слід мати відпрацьований IR-playbook для випадків підозри на руткіт (ізоляція, форензика пам'яті і диска, координація з виробником/постачальником, відновлення з довірених образів) і механізми обміну threat intelligence з галузевими партнерами та правоохоронцями.

План реагування на інцидент із руткітом: злагоджені дії від виявлення до відновлення

Ефективне реагування на інцидент із руткітом починається з ретельної підготовки: визначення ролей (IR-лідер, SOC/triage, Forensics, IT/Ops, Network/NOC, malware-аналітик, юридичний відділ, PR), тестування процедур збору доказів і наявності довірених резервних копій, встановлення контактів із виробниками обладнання та CERT/CSIRT, а також підтримки апаратно-орієнтованих механізмів захисту (Secure Boot, TPM, HVCI). Без попередньої підготовки успішне виявлення, ізоляція та довготривале видалення руткіта значно ускладнюються, тому на етапі «Preparation» потрібно мати задокументовані playbook-и, процедури chain-of-custody і доступ до інструментів пам'яті та прошивкової форензики (memory forensics, UEFITool/Chipsec, інструменти для порівняння образів ядра).

Виявлення зазвичай починається з аномалій у поведінці системи (приховані процеси, відсутність очікуваних записів у списку процесів, невідповідності контрольних сум ядра, дивні відповіді інструментів діагностики) або з мережових маячків (нестандартні beacon-патерни, незвичні DNS-запити). Після первинного сигналу SOC виконує швидкий триаж: підтверджує наявність індикаторів компрометації, збирає базову телеметрію (EDR/AV, системні логи, Sysmon) і повідомляє IR-лідера; якщо є підозра на руткіт, команда негайно активує режим форензики і containment, щоб не допустити поширення та не знищити артефакти.

Containment за руткітом має бути виваженим: просте перезавантаження або спроби «лікування» можуть стерти пам'ятні артефакти і знищити докази, необхідні для розуміння механізму інтеграції в систему. Тож перший крок — ізоляція уражених вузлів від мережі (EDR-isolate, VLAN/ACL, фізичне відключення), при

цьому не виконуючи дій, що змінюють стан пам'яті до моменту зняття пам'ятного дампу. Паралельно слід обмежити адміністративний доступ (MFA, тимчасова відміна віддалених сесій, блокування скомпрометованих облікових записів) і впровадити мережеві блокування для виявлених C2-патернів. Для серверів/контролерів домену containment може вимагати фази — ізоляція, збереження образів, після чого — переключення на резервну інфраструктуру.

Критично важливим є збирання доказів: перш ніж будь-що змінювати, команда форензики робить memory dump (знімок оперативної пам'яті), форензичний клон диска (bit-stream image), зберігає журнали системних подій, Sysmon, NDR/PCAP, а також копії прошивок пристроїв (UEFI/BIOS, контролери дисків). Всі дії фіксуються в chain-of-custody з часовими мітками і підписами. Аналіз пам'яті (Volatility/Rekall) дозволяє виявити інлайн-хуки, завантажені ядрові модулі, незвичні структури процесів (DKOM), тоді як UEFITool/Chipsec і Binwalk застосовуються для дослідження прошивок і виявлення модифікацій на низькому рівні.

Після первинного аналізу слід визначити scope-інфекції: які системи під контролем, чи є зміни в завантажувачі (MBR/UEFI), чи вражені прошивки пристроїв (HDD, NIC, RAID-контролери) і чи скомпрометовано контролери домену. Якщо руткіт вкорінився в UEFI/прошивці або на рівні контролерів, стандартні процедури «видалення» зазвичай неефективні — може знадобитись перевишивка/перепрограмування прошивки або навіть апаратна заміна компоненту. Рішення про заміну пристрою повинно прийматись з урахуванням результатів форензики і у взаємодії з виробником обладнання.

Етап усунення починається після документованого розуміння механізму проникнення і переліку заражених артефактів: видаляються/блокуються виявлені непідписані драйвери, відкатуються змінені бінарі (де це безпечно), видаляються або перевстановлюються служби автозапуску, масово міняються паролі і впроваджується MFA для всіх адміністративних облікових записів. Якщо виявлено модулі, що працюють у пам'яті (fileless), виконують ін'єкції або hook-и ядра, необхідно використати спеціалізовані інструменти пам'яттєвої форензики і попрацювати у співпраці з виробниками EDR/OS-постачальниками для коректного видалення. У разі компрометації AD рекомендується провести аудит привілеїв, ротацію ключів і, за потреби, rebuild контролерів з довірених бекапів.

Відновлення має базуватися на перевірених, чистих образах. Після очищення і патчингу систем виконують поетапне повернення в експлуатацію: спочатку тестове відновлення критичних сервісів в ізольованому середовищі з увімкненим розширеним моніторингом, далі — поступове повернення в production з огляданням на повторні сигнали. Якщо була вражена прошивка або UEFI і виконано перевишивку або заміну апаратного блоку, перед повторним введенням у виробництво необхідна подвійна перевірка цілісності (контрольні суми, підписи, апаратні виміри — TPM/Measured Boot).

Після відновлення слідує постінцидентний аналіз: детальний звіт з таймлайном, root-cause, list IOC, оцінка втрат і ефективності реагування, оновлення IR-playbook-a і процедур. Також необхідно оновити правила моніторингу (SIEM, EDR, NDR), додати виявлені IOC у блок-листи і поділитися релевантною розвідкою з партнерами та CERT/ISAC, зберігаючи при цьому юридично необхідні обмеження щодо розголошення. Організаційні заходи включають підвищення привілеїв безпеки (application whitelisting, HVCI), регулярні аудити прошивок, політику захищених оновлень і контроль ланцюга постачання ПЗ (SBOM).

Кожен крок повинен супроводжуватися процедурою комунікації: внутрішні оновлення для керівництва і технічних команд, юридичні консультації щодо повідомлення регуляторів і клієнтів, а також підготовка PR-заяви у разі необхідності. Для майбутнього важливо провести навчання персоналу, провести tabletop-вправи і перевірити готовність до сценаріїв з rootkit-типом загроз.

Короткий чек-ліст першої години: 1) підтвердити інцидент і активувати IR-групу; 2) ізолювати підозрілі вузли (EDR-isolate або фізичне відключення) без перезавантаження; 3) зняти дампи пам'яті і форензичні образи дисків; 4) блокувати відомі C2/DGA-IOC на DNS/firewall; 5) повідомити виробників обладнання і CERT; 6) зафіксувати chain-of-custody. Метрики успішності реагування (TTD, TTC, TTR, кількість уражених хостів, наявність скомпрометованих прошивок) допомагають оцінити якість виконаних дій і планувати поліпшення.

Такий підхід — поєднання підготовки, акуратної ізоляції, суворої форензики, співпраці з виробниками й правоохоронцями, а також системного відновлення з довірених джерел — дає найбільші шанси повністю нейтралізувати руткіт і відновити довіру до інфраструктури.

Ботнети

Ботнети, у свою чергу, являють собою мережі заражених пристроїв, які підконтрольні зловмисникам і використовуються для здійснення масштабних атак. Їхня сила полягає у розподіленій природі: до складу ботнету можуть входити сервери, персональні комп'ютери та IoT-пристрої. Вони часто застосовуються для проведення DDoS-атак, масової розсилки спаму, організації фішингових кампаній чи прихованого майнінгу криптовалют.

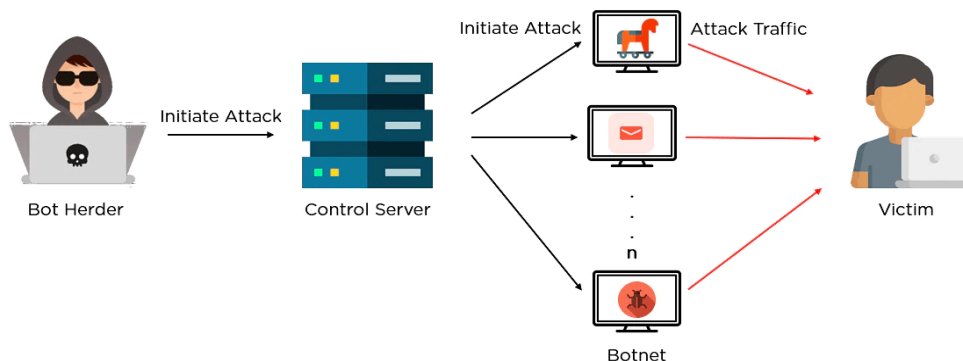


Рис. 2.8. Схема роботи ботнета

Джерело: [1-20, 25-34, 42-56]

Структура ботнету — це багаторівнева архітектура, яка забезпечує інфікованим пристроям координацію, стійкість і можливість виконувати централізовані або розподілені команди; розуміння її компонентів важливе для ефективного виявлення та знешкодження. У центрі архітектури — самі боти (зомбі) — інфіковані вузли (сервери, ПК, IoT-пристрої), що виконують завдання, отримані від керівника мережі; на ботах працюють модулі персистентності, модулі реалізації завдань (DDoS, спам, майнінг, ексфільтрація) і механізми оновлення. Командно-контрольна інфраструктура (C2) забезпечує зв'язок між оператором і ботами і може бути реалізована як централізована (IRC, HTTP/HTTPS-сервери, REST-API), децентралізована (P2P-протоколи, де кожен бот містить список пір) або гібридна з резервними каналами зв'язку.

Для підвищення стійкості ботнети застосовують механізми обходу блокувань: алгоритми генерації доменів (DGA) для знаходження C2-доменів, fast-flux і double-flux для динамічної маскуваності DNS-записів, а також використання проксі-серверів і TOR-мережі як каналу зв'язку. Часто у складі є завантажувачі/дропери, які забезпечують первинне проникнення й доставку основних модулів, а також модулі самозахисту — шифрування трафіку, обфускація, перевірка середовища виконання та анти-форензичні трюки. Адміністративний рівень включає панель управління (web-панель або CLI), бази даних операторів, платіжні та монетизаційні компоненти (оренда ботнету, продаж доступу), а також логіку розподілу завдань і механізми звітування про виконання. На мережевому рівні ботнети мають списки пір/реплікаторів, механізми вибору лідера або fallback-сервера, таблиці черг завдань і таймери «контрольного маячка» (beaconing) для періодичного зв'язку з C2.

Для латерального руху і розширення бази ботів використовуються експлойти, слабкі паролі, brute-force, соціальна інженерія та компрометація ланцюгів постачання — ці механізми інтегруються в логіку розподілу завдань ботнету. З точки зору захисту, критичними точками структури є C2-сервери, доменні і DNS-інфраструктури (DGA/fast-flux), панелі управління і маршрути ексфільтрації; їхня виявлення й перехоплення (sinkholing) часто призводять до зупинки операцій.

Розуміння цих компонентів дозволяє проєктувати заходи: моніторинг аномалій у трафіку і beaconing-патернів, аналіз DNS-запитів і DGA-сигнатур, кореляцію IOC у SIEM, блокування доменів та IP, а також координацію з провайдерами і правоохоронцями для takedown-операцій.

Додатково ефективними заходами є розгортання honeypot-інфраструктур і sinkhole-серверів для вивчення поведінки ботів та отримання індикаторів компрометації, що підсилює threat intelligence і прискорює блокування інфраструктури противника. Важливо посилювати безпеку кінцевих пристроїв через регулярне патчування, жорсткі політики паролів, багатофакторну автентифікацію та застосування принципу найменших привілеїв, особливо для адміністративних інтерфейсів (RDP, SSH, панелі керування). На мережевому рівні допомагають сегментація, фільтрація трафіку за поведінковими сигнатурами, rate-limiting та використання NDR/IDS для швидкого виявлення аномалій, а також автоматизовані playbook-и для оперативного containment і remediation. Довгострокова протидія вимагає міжнародної координації, обміну розвідданими між операторами, постачальниками та правоохоронцями, а також заходів щодо підвищення безпеки ланцюгів постачання і стандартизації захищених налаштувань для IoT-виробів.

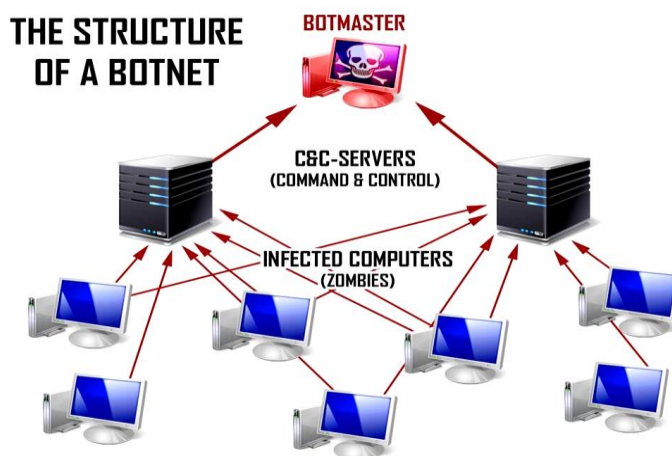


Рис. 2.9. Структура ботнету: ботмастер, сервери командного управління (C2) та інфіковані «зомбі»-вузли

Джерело: [1-20, 23-30, 35-48]

Ботнети також характеризуються наявністю командно-контрольної інфраструктури (C2), через яку оператори видають інструкції зараженим вузлам і координують їхні дії. Деякі сучасні ботнети використовують децентралізовані P2P-протоколи для підвищення стійкості та ускладнення виведення з ладу шляхом ліквідації центрального сервера. Великий розмір і географічне розсіяння ботнету дозволяють зловмисникам масштабувати атаки та обходити локальні заходи захисту, одночасно приховуючи джерело активності. Оператори монетизують ботнети через

надання DDoS-послуг «за підпискою», оренду мережі для розсилок або продаж доступу іншим злочинним групам на підпільних ринках. Уразливі налаштування пристроїв Інтернету речей і слабкі або стандартні паролі залишають величезну площину для набору нових ботів, що робить IoT-екосистему особливо привабливою ціллю. Виявлення ботнет-активності базується на аналізі аномалій у мережевому трафіку, кореляції підозрілих зв'язків із відомими індикаторами компрометації та використанні NDR/IDS-систем.

Методи нейтралізації включають sinkholing доменів C2, блокування IP-адрес і доменів, координацію з провайдерами хостингу та правоохоронними органами для виведення інфраструктури з експлуатації. Ефективна протидія вимагає також оновлення прошивок, застосування сильних аутентифікаційних механізмів і сегментації мережі, щоб ускладнити латеральний рух і розповсюдження в межах корпоративної інфраструктури. Керована і прозора співпраця між постачальниками інтернет-послуг, виробниками обладнання та органами безпеки є критичною для швидкого виявлення і знешкодження ботнетів в інтернет-масштабі. З огляду на комерціалізацію кіберзлочинності, прогнозовано, що ботнет-екосистема залишатиметься активною і еволюціонуватиме, вимагаючи постійного оновлення детекційних та превентивних механізмів.

Класифікація ботнетів охоплює кілька взаємопов'язаних вимірів — архітектурного, функціонального, цільового та операційного — кожен із яких визначає їхню поведінку, стійкість і підходи протидії. За архітектурою розрізняють централізовані ботнети з одним або кількома C2-серверами (IRC/HTTP), децентралізовані P2P-мережі, де кожен бот може виступати і як клієнт, і як ретранслятор, а також гібридні моделі, що поєднують обидва підходи для підвищення стійкості. За призначенням ботнети діляться на ті, що орієнтовані на проведення DDoS-кампаній, розсилку спаму та фішингових повідомлень, організацію прихованого майнінгу криптовалют, створення проксі-мереж для анонімізації трафіку, викрадення облікових даних і масштабні кампанії з ексфільтрації даних або виконання фінансових транзакцій через компрометовані системи. За типом цілі виділяють IoT-ботнети, що залучають уразливі пристрої зі слабкими паролями, корпоративні ботнети, орієнтовані на сервери й кінцеві робочі станції, та мобільні мережі, що атакують смартфони та планшети. За механізмом розповсюдження ботнети класифікують як саморозповсюджувані (через експлойти та черв'ячні механізми), соціально-інженерні (через фішинг і підроблені оновлення) або комбіновані, що використовують багатоканальну інфраструктуру проникнення. За стійкістю й маскуванню деякі ботнети застосовують DGA/fast-flux/DNS-обфускацію і проксі-шари для приховування C2, інші ж роблять ставку на шифровані P2P-канали або TOR-тунелювання. Окрема класифікація торкається комерційної моделі: «ботнети як послуга» (BaaS), орендовані мережі та приватні ботнети для внутрішніх АPT-операцій; це визначає рівень доступності інструментів і швидкість

еволюції загроз. І нарешті, за тривалістю і керованістю розрізняють тимчасові (кампанійні) ботнети, створені для однієї великої акції, і довготривалі мережі з відпрацьованою інфраструктурою монетизації. Усвідомлення цих класів не лише допомагає краще ідентифікувати ІОС і патерни поведінки, а й формувати прицільні заходи захисту — від сегментації мережі й жорстких політик аутентифікації до DNS-фільтрації, sinkholing-операцій і міжнародної координації для takedown-операцій.

План реагування на інцидент із ботнетом: згаджені дії від виявлення до відновлення

План описує комплекс заходів для виявлення, локалізації, нейтралізації та відновлення після інциденту, пов'язаного з ботнет-інфікуванням — від початкового триажу до координації takedown-операцій з провайдерами та правоохоронними органами. Мета заходів — мінімізувати масштаби зараження, перервати зв'язок елементів ботнету з С2-інфраструктурою, відновити контроль над інфраструктурою та зменшити ризики повторної компрометації й ексфільтрації даних.

Підготовка — ключовий етап успішного реагування. До початку інциденту необхідно чітко визначити ролі й канали зв'язку: IR-лідер для координації, SOC/triage для детекції, Network/NOC для мережевих правил, IT/Ops для ізоляції й відновлення, Forensics/Malware Analysts для збору й аналізу артефактів, Legal/Compliance і PR для юридичної та публічної взаємодії, а також мати заздалегідь контакти провайдерів, CERT/CSIRT і правоохоронців. Потрібні тестовані playbook-и, процедури chain-of-custody та довірені резервні копії конфігурацій і даних — усе це скорочує час реагування і підвищує якість ухвалених рішень у кризі.

Виявлення та первинний триаж починаються з моніторингу телеметрії: масова аномалія в мережевому трафіку (регулярне beaconing), підвищене навантаження на зовнішні канали, хвилі спаму, DDoS-активність з внутрішніх вузлів або повідомлення користувачів можуть свідчити про ботнет. SOC проводить швидку оцінку, підтверджує наявність ІОС (IP/DNS, hashes, mutex), масштабує перелік уражених об'єктів (ендпоінти, сервери, IoT) і встановлює архітектуру С2 (централізована чи P2P). Підтвердження інциденту — сигнал до активації IR-команди й початку containment.

Першочергова задача containment — локалізувати зараження і зупинити розповсюдження. Це досягається через EDR-ізоляцію хостів, тимчасове обмеження мережевого доступу (VLAN/ACL), блокування відомих С2-доменів і IP на рівні DNS і firewall та, за потреби, відключення поштових релєїв чи інших каналів розповсюдження. Для IoT/OT-пристроїв застосовують сегментацію мережі й локальні ACL, щоб уникнути вторинного зараження. Всі зміни мають координуватися з NOC: containment не повинен необґрунтовано зруйнувати критичні сервіси.

До виконання агресивних дій слідує етап збору доказів і форензики. Перед будь-якими змінами необхідно зробити знімки пам'яті і форензичні образи дисків

уражених машин, зберегти PCAP і журнали (проху, firewall, VPN), а також телеметрію EDR/IDS і конфігурації мережевого обладнання. Forensics-команда аналізує зразки для виявлення механізмів C2 (протокол, DGA, P2P), шляхів розповсюдження (експлойти, compromised credentials) і формує перелік IOC для розгортання блокувань; усі дії документуються в chain-of-custody.

Мережеві контрзаходи та тісна співпраця з провайдерами є критично важливими для масштабних ботнетів. На базі IOC негайно застосовуються блокування на firewall/DNS/IPS і оновлюються правила IDS/NDR для кореляції. Координація з ISPs, хостинг-провайдерами і реєстраторами доменів дозволяє організувати sinkholing, блокування або takedown C2-серверів; IR-лідер має мати заздалегідь процедури запиту на takedown і налагоджені канали комунікації з провайдерами, CERT/CSIRT і правоохоронцями.

Після блокувань переходять до видалення і відновлення контролю над інфікованими пристроями: масова очистка кінцевих точок (видалення шкідливих бінарів), оновлення або перевстановлення вразливих компонентів, зміна скомпрометованих облікових даних і застосування патчів. Для IoT-пристроїв, які не підтримують надійне очищення, може знадобитися перевипуск прошивки або фізична заміна. Відновлення сервісів має виконуватися з перевірених чистих образів та супроводжуватися жорсткішими політиками доступу і сегментацією, щоби знизити ризик повторного проникнення.

Після containment і початкового відновлення встановлюють розширений період моніторингу: підвищений рівень логування, детальний аналіз аномалій, перевірка на повторні beacon-патерни і кореляція нових IOC. Проводиться аудит змін конфігурації і перевірка цілісності образів. Довгострокові заходи включають оновлення IDS/EDR-сигнатур, підтримку актуальних блок-листів DNS і розгортання митигуючих правил у хмарних та IoT-середовищах.

Юридичні, регуляторні та комунікаційні аспекти повинні супроводжувати технічні дії: оперативне інформування керівництва, Legal і, за вимогами, регуляторів і клієнтів; PR-комунікація має бути скоординована і контролюватися для мінімізації репутаційних ризиків. Юридичний супровід допомагає визначити вимоги щодо збереження доказів, повідомлення про витік даних і взаємодії з правоохоронцями.

Після завершення операцій проводиться постінцидентний аналіз: after-action review із документуванням timeline, root-cause, оцінкою економічних втрат і ефективності containment-заходів. На основі висновків оновлюються playbook-и, політики безпеки, мережеві конфігурації й процеси управління постачальницькими ризиками; проводяться навчання персоналу, red-team/blue-team вправи та тестування процедур takedown і sinkholing.

Для вимірювання ефективності реагування важливо відстежувати метрики: Time to Detect (TTD), Time to Contain (TTC), кількість інфікованих пристроїв, Time to Recover (TTR), обсяги ексфільтрованих даних (якщо були) та кількість успішних

takedown-операцій. Швидкий чек-ліст першої години: підтвердити інцидент і активувати IR-команду; ізолювати уражені вузли (EDR/isolate); блокувати відомі C2-IOC на DNS/firewall; почати збір memory/disk/PCAP; повідомити провайдерів і CERT; призупинити автоматичні оновлення резервних копій, щоб не знищити артефакти.

Узгодженість технічних дій, чітка форензична дисципліна, швидка координація з провайдерами і правоохоронцями, а також поєднання короткострокових мережових блокувань з довгостроковими організаційними змінами — ось що гарантує ефективне припинення діяльності ботнету і стійке відновлення інфраструктури.

Програми-вимагачі

Нарешті, ransomware, або програми-вимагачі, стали однією з найбільш небезпечних форм загроз за останні роки. Вони блокують доступ до системи або шифрують дані користувача, вимагаючи викуп за відновлення доступу. Сучасні варіації використовують сильні криптографічні алгоритми, що робить практично неможливим відновлення даних без ключа розшифрування. До того ж, поширеною стала практика «подвійного шантажу», коли перед шифруванням зловмисники копіюють дані й погрожують оприлюднити їх у випадку відмови від сплати.

У багатьох випадках операції з ransomware супроводжуються ретельною розвідкою та латеральним переміщенням у мережі задля максимального охоплення критичних ресурсів перед шифруванням. Окремі угруповання працюють за моделлю RaaS (Ransomware-as-a-Service), коли розробники продають чи здають в оренду інфраструктуру кураторам-афіліатам, що прискорює масштабування атак. Публікація викрадених даних на «шантажних» сайтах і торги за знищення витоків створюють додатковий репутаційний і юридичний тиск на постраждалих, особливо за наявності персональних даних, що регулюються законами про захист даних. Оплата викупу не гарантує повного відновлення даних і часто заохочує подальші злочинні дії, тому політика реагування повинна включати співпрацю з правоохоронними органами та фахівцями з реагування на інциденти. Найефективніші контрзаходи — регулярне тестоване резервне копіювання, сегментація мережі, обмеження привілеїв і своєчасне патчування вразливостей разом із проактивним моніторингом і планом відновлення бізнес-процесів. Додатково варто впроваджувати превентивні процедури (DLP, шифрування даних у спокої, MFA) та навчання персоналу, щоб зменшити ймовірність початкового проникнення через фішинг або компрометацію облікових записів.

Класифікація програм-вимагачів охоплює кілька взаємопов'язаних вимірів, що допомагають зрозуміти їхню природу, способи роботи й підходи до протидії. За функціональною ознакою традиційно розрізняють крипто-вимагачі (crypto-ransomware), які шифрують файли та каталоги жертви, локер-вимагачі (locker), що блокують доступ до системи або інтерфейсу без масового шифрування, і wiper-

подібні модулі, які маскуються під ransomware, але фактично безповоротно руйнують дані. За методикою вимагання виділяють одинарний шантаж (шифрування + вимога викупу), подвійний шантаж (копіювання даних перед шифруванням і погроза витоку) та розширені схеми з трипл-шантажем (публікація даних, DDoS проти постраждалих та вимога викупу). За моделлю операційної організації атаки бувають комерційні RaaS (Ransomware-as-a-Service) з відокремленими розробниками та афіліатами, і цілеспрямовані АРТ-кампанії, де групи самостійно виконують розвідку, латеральний рух і індивідуальне шифрування критичних ресурсів. За каналами розповсюдження виділяють вектори через фішинг і шкідливі вкладення, експлуатацію відкритих RDP/адмін-інтерфейсів, компрометацію ланцюгів постачання і використання початкових завантажувачів/дропперів.



Рис. 2.9. Повідомлення програми-вимагача (ransomware): екран із вимогою викупу
Джерело: [1-20]

Платформенно-технологічна класифікація враховує орієнтацію на Windows, Linux, macOS, контейнерні і хмарні середовища, а також мобільні платформи; це визначає інструменти шифрування, маршрути ескалації та методи відновлення. За криптографічною схемою розрізняють реалізації з симетричним шифруванням, асиметричним або гібридним підходом (симетричний ключ зашифровано публічним ключем атакуючого), що впливає на можливість розшифрування без участі зловмисників. За рівнем складності — від масових «commodity» версій із шаблонними механізмами до модульних, поліморфних і fileless-варіантів із сильними анти-форензичними техніками; останні часто поєднують ексфільтрацію даних із шифруванням і тривалою персистентністю.

Окремо виділяють класи за комерційною метою: «орендовані» бот-мережі для DDoS/шантажу, фабрично створені мережі для прихованого майнінгу та ботнет-інтегровані схеми. Розуміння цих класів допомагає визначати індикатори компрометації (IOC) і підбирати заходи захисту — від регулярного тестованого резервного копіювання, сегментації й MFA до поведінкової аналітики, EDR/NDR і процедур incident response з блокуванням ексфільтрації й sinkholing-операціями. На практиці ефективна стратегія протидії повинна враховувати одночасно кілька вимірів класифікації, оскільки сучасні кампанії поєднують архітектурні, криптографічні та організаційні елементи.

Таблиця 2.4. Класифікація програм-вимагачів (ransomware): тип, характеристика, приклади, методи захисту

Тип	Характеристика	Приклади	Методи захисту / рекомендації
Crypto-ransomware (криптор)	Шифрує файли/каталоги жертви; зазвичай використовує сильну криптографію; вимагає викуп за ключ.	WannaCry, CryptoLocker, Ryuk	Регулярні тестовані резервні копії, сегментація мережі, MFA, патчинг, EDR, швидкий інцидент-респонс.
Locker (локер)	Блокує доступ до системи або інтерфейсу (без масового шифрування файлів).	WinLocker-сімейства	Забезпечення доступу до адміністрування, резервні образи систем, політики доступу, антивірус/EDR.
Wiper (імітує ransomware)	Виглядає як ransomware, але безповоротно руйнує/стирає дані (кіберсаботаж).	NotPetya, Shamoon	Ізоляція інцидентів, офлайн-резервні копії, контроль цілісності, співпраця з правоохоронцями.
Double extortion (подвійний шантаж)	Перед шифруванням ексфільтрують копію даних; вимагають викуп і за нерозголошення.	Maze, DoppelPaymer	DLP, шифрування даних у спокої, обмеження ексфільтрації, швидке виявлення аномалій, юридична підготовка.
Triple / multi-extortion	Поєднує шифрування + витік + DDoS/шкода репутації/тиск третіх сторін.	Ряд сучасних кампаній	Комплексний план реагування, резервні копії, підготовка комунікаційної стратегії, DDoS-мітігація.
RaaS (Ransomware-as-a-Service)	Модель комерційного надання платформи: розробник + афіліати; швидке масштабування атак.	REvil (Sodinokibi), Conti (моделі)	Моніторинг фінансових каналів, threat intelligence, блокування відомих інфраструктур, індикатори IOC.

Тип	Характеристика	Приклади	Методи захисту / рекомендації
Targeted / APT-орієнтований ransomware	Ретельна розвідка, LAT-рух (латеральний), орієнтація на критичні активи організації.	Ryuk (APT-кампанії), DarkSide	Сегментація критичних мереж, MFA для адміністраторів, строгий контроль привілеїв, активний threat hunting.
Fileless / stealthy ransomware	Працює в пам'яті або через легітимні інструменти (LOTL), мінімізує сліди на диску.	Певні варіанти GandCrab/Conti модулів	EDR з моніторингом пам'яті, аналіз поведінки процесів, контроль PowerShell/Script-execution, обмеження адмін-доступів.
Cloud / container ransomware	Орієнтований на SaaS/облікові записи/контейнери → шифрування даних у хмарі.	Модульні кампанії проти хмарних сервісів	MFA для облікових записів, контроль доступів API, аудит логів, резервні копії хмарних даних, least-privilege для сервісних акаунтів.
Mobile ransomware	Блокує або шифрує дані мобільних пристроїв; часто через шкідливі додатки.	Android-варіанти (Locker), прості сімейства	Політика встановлення додатків, мобільні EDR, контроль дозволів, навчання користувачів.
Commodity / mass-spam ransomware	Масові розсилки; менш складні, але спричиняють широкі втрати у кількості жертв.	Petya-клони, ранні версії Cryakl	Фільтрація пошти, sandboxing вкладень, користувацьке навчання, сегментація мережі.
Hybrid / modular (модульні)	Має компоненти для ексфільтрації, шифрування, lateral movement; легко оновлюється.	TrickBot + Ryuk взаємодія	Комплексні SIEM/EDR + NDR, кореляція IOC, оперативний incident response, резервні копії.

Таким чином, класифікація шкідливого програмного забезпечення охоплює широкий спектр загроз — від традиційних вірусів і троянів до складних ботнетів і програм-вимагачів. Кожна з цих категорій має специфічні механізми дії, але їх об'єднує спільна мета — порушення конфіденційності, цілісності та доступності інформації. Усвідомлення їхніх характеристик і моделей функціонування є основою для побудови багаторівневих систем захисту, які поєднують технічні, організаційні та освітні заходи. Це дозволяє не лише вчасно виявляти та нейтралізувати атаки, а й формувати довгострокову стратегію кіберстійкості підприємств.

Програма-вимагач звичайно побудована як модульна, взаємопов'язана система, що поєднує набори компонентів для доставки, закріплення, розширення контролю, шифрування даних і монетизації атаки; процес починається з

початкового вектору — фішингового листа, шкідливого вкладення або скомпрометованого сервісу, через які в систему жертви потрапляє легкий дроппер або загрузчик, задача якого — доставити і підготувати виконання основних модулів.

Далі звичайно працює стартовий loader, який виконує базові перевірки середовища, мінімізує свій розмір і встановлює персистентність — запис у автозавантаження, створення служб або завдання планувальника, інколи більш стійкі механізми, що дозволяють пережити перезавантаження або приховано підвантажити додаткові компоненти.

Одночасно або після закріплення запускаються модулі ескалації привілеїв і латерального руху, які збирають облікові дані, використовують легітимні адміністративні інструменти та вбудовані сервіси для переміщення всередині мережі; ці механізми дають змогу зловмисникам розширити охоплення мережі й ідентифікувати критичні ресурси для цілеспрямованого шифрування.

Часто перед шифруванням виконується ексфільтрація — копіювання важливих даних назовні для подальшого шантажу, що робить так званий подвійний шантаж особливо руйнівним для організацій, адже загрожує не лише втратою доступу, а й компрометацією конфіденційної інформації.

Серце програми-вимагача — модуль шифрування або локер, який може працювати за симетричною, асиметричною або гібридною схемою шифрування; він прагне зашифрувати максимально велике число цільових файлів, змінюючи їхні розширення й створюючи файли-інструкції для жертви з вимогами щодо викупу. Зазвичай шифрувальний модуль генерує випадкові симетричні ключі для ефективного шифрування великих обсягів файлів і зберігає їх у конфігурації, після чого ці симетричні ключі самі шифруються публічним ключем оператора (гібридна схема), що унеможлиблює відновлення без приватного ключа нападника. Реалізація часто включає селекцію цільових типів файлів за масками і розширеннями, багатопотокове виконання для прискорення шифрування, попереднє завершення критичних сервісів чи процесів і спроби знищити або пошкодити локальні/тонкі резервні копії (наприклад видалення теней томів), щоб ускладнити відновлення. Після завершення операцій модуль залишає інструкції для жертви (ransom-note) з адресою для зв'язку через анонімні канали (Tor-сайти, крипто-гаманці) і часто супроводжується ексфільтрацією даних для подальшого шантажу (подвійний шантаж). Операційні індикатори такого поведінкового патерну — раптове зростання файлової активності/ІО, масова зміна імен файлів або розширень, поява файлів-сповіщень і команд для видалення теней томів — і їхнє своєчасне виявлення (EDR/behavioral detection, блокування процесів з незвичним масовим доступом до файлів, наявність незмінних офлайн-бекапів та сегментація мережі) суттєво підвищують шанси на мінімізацію втрат.

Управління та зв'язок реалізуються через інфраструктуру командно-контрольних каналів (C2) — централізовані сервери, DGA-домени, P2P або анонімні мережі — які забезпечують отримання команд, оновлення модулів і передачу ключів; до них проявляють увагу засоби мережевого моніторингу через виявлення маячків і аномальних DNS/IP-патернів.

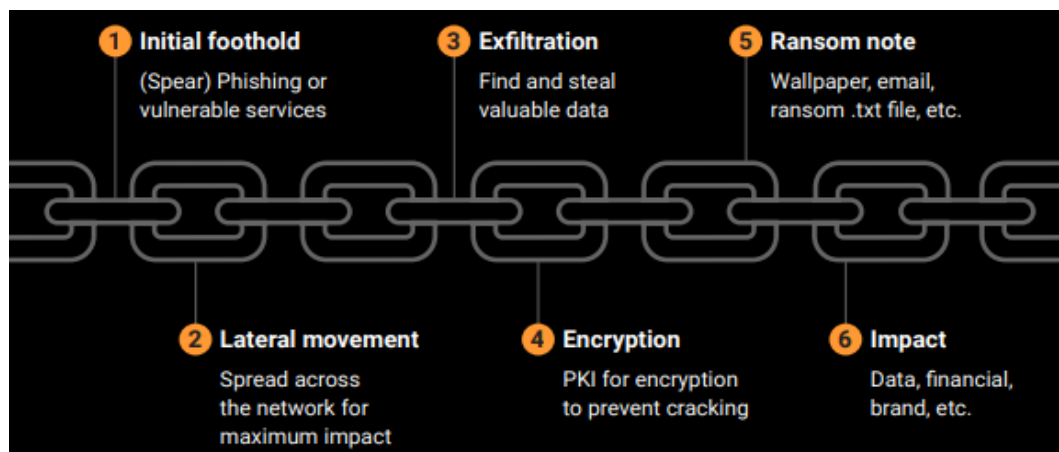


Рис. 2.10. Схема «kill-chain» для атак ransomware: 1) первинне проникнення; 2) латеральний рух; 3) ексфільтрація; 4) шифрування; 5) повідомлення-вимога; 6) наслідки (втрата даних, фінансові та репутаційні збитки)
Джерело: [1-20, 25-34]

Для максимального виживання зразки використовують механізми антидетекції та антифореңзики — обфускацію коду, шифрування конфігурацій, перевірки середовища виконання, fileless-підходи та інші трюки, які ускладнюють аналіз і видалення; багато сімейств також мають модуль оновлення або плагінну архітектуру, що дозволяє динамічно додавати функціонал (ексфільтрацію, майнінг, проксі) без повторної первинної доставки. Багато сімейств реалізують модульні оновлення та зловживання цифровими підписами — через це критично збирати процесну й пам'ятну телеметрію з EDR (Endpoint Detection and Response), логи SIEM (Security Information and Event Management) і мережеві захоплення (PCAP) для кореляції підозрілих оновлень модулів із зовнішніми маячками.

Detection engineering повинен включати YARA-правила для бінарів і шаблонів у пам'яті, Sigma-правила для уніфікованого опису детекцій у SIEM, кластеризацію поведінкових звітів зі sandbox і регулярну валідацію детекцій через memory-forensics (аналіз дамів пам'яті). Профілактичні заходи — обов'язкове підписування і перевірка цифрових підписів для критичних бінарів, application control / whitelisting, принцип найменших привілеїв (least privilege) і мікросегментація мережі — значно ускладнюють можливість динамічного дозавантаження чужих плагінів або lateral movement. Нарешті, ефективна оборона поєднує технічні контрзаходи з операційними практиками: регулярний обмін threat-

intelligence (розвіданими про загрози), періодичні red/purple-team вправи для тестування детекцій і playbook-ів та підтримка незмінних (immutable) офлайн-резервних копій для швидкого відновлення після компрометації.

Індикаторами компрометації можуть слугувати раптові масові зміни файлів, поява ransom-note, незвичні вихідні з'єднання, нові служби або записи автозавантаження, а також аномалії в логах аутентифікації — їхнє відстеження разом із кореляцією IOC у SIEM та NDR/EDR дає найкращі шанси на раннє виявлення.

План реагування на інцидент із програмою-вимагачем (ransomware): злагоджені дії від виявлення до відновлення

Захист проти програм-вимагачів базується на багаторівневому підході: превентивні заходи (жорсткі політики привілеїв, MFA, application whitelisting, сегментація мережі), надійні та тестовані резервні копії (offline/immutable), регулярний патчинг і контроль конфігурацій, а також сучасні засоби виявлення — EDR з моніторингом пам'яті, NDR для поведінкової аналітики, автоматизовані playbook-и для containment і процес інцидент-респонсу зі сценаріями відновлення.

Ефективне реагування на інцидент із програмою-вимагачем починається задовго до появи першої ознаки компрометації — з підготовки інфраструктури, політик і ролей. Важливо заздалегідь визначити команду реагування (IR-лідер, SOC/triage, IT/Ops, forensic-аналітик, мережеві операції, юридичний відділ та PR), забезпечити контакти зовнішніх постачальників і правоохоронців, протестовані резервні копії (offline/immutable) та механізми моніторингу (EDR, NDR, SIEM). Наявність готових playbook-ів, каналів зв'язку й відпрацьованих процедур дозволяє скоротити час виявлення й прийняття рішень в момент інциденту.

Після отримання тривожного сигналу віддетекторів чи користувачів SOC проводить первинну триаж-оцінку, перевіряє наявність ransom-note, масштаб шифрування та ознаки beacon-комунікацій. Якщо інцидент підтверджено, IR-лідер активує команду, повідомляє керівництво й юридичний підрозділ, після чого першочерговим завданням стає containment — ізоляція уражених хостів від мережі (через EDR або фізично), тимчасове блокування зовнішнього доступу та обмеження адміністративних каналів, щоб не допустити подальшого розповсюдження та ексфільтрації даних.

Паралельно з ізоляцією необхідно зібрати та зберегти докази: зробити знімки оперативної пам'яті, створити форензичні іміджі дисків, зберегти логи подій, мережеві захоплення (PCAP) і конфігурації пристроїв. Ці артефакти критично важливі для подальшого аналізу та можливого співробітництва з правоохоронцями, тому всі дії фіксуються в chain-of-custody. Форензик-команда і malware-аналітики аналізують зразки, витягують IOC (хеші, mutex-и, домени C2), перевіряють наявність інлайн-хуків чи нестандартних драйверів і корелюють знахідки у SIEM/NDR для визначення повного охоплення компрометації.

Коли вектор проникнення, методи і score інциденту виявлені, команда впроваджує заходи повного containment: блокує C2-сервери, sinkhole-ить DGA-домен, відключає скомпрометовані облікові записи та застосовує блокуючі правила на мережевих і прикладних рівнях. Рішення про ймовірну оплату викупу приймається на рівні власників бізнесу за участі юридичного відділу й правоохоронних органів; у більшості випадків рекомендується уникати виплат, проте остаточне рішення має бути документованим і оціненим за ризиками.

Етап усунення передбачає повне видалення шкідливих компонентів і руйнування бекдорів: ревізію автозавантажень, служб і планувальників, видалення непідписаних драйверів та змін у AD, а також масову ротацію паролів і впровадження MFA. Якщо домен контролерів чи прошивки були скомпрометовані, може знадобитись їхнє перебудування або перевишивка пристроїв. Важливим підходом є не намагатися «лікувати» окремі машини, а відновлювати сервіси з довірених, перевірених резервних копій у контрольованому середовищі.

Повернення до нормальної роботи здійснюється поетапно: спочатку відновлюються критичні сервіси у тестованому ізольованому середовищі з посиленням моніторингом, потім — залежні системи, і лише після повної перевірки — повне повернення в продакшен. Під час і після відновлення потрібно зберігати підвищений рівень уваги в monitoring-вікні, щоб упевнитися у відсутності повторної активності або прихованих модулів. Необхідно фіксувати та архівувати всі судово-експертні артефакти (дампи пам'яті, образи дисків, журнали подій, мережеві дампи) із забезпеченням коректного ланцюга зберігання доказів для подальшого аналізу та можливих юридичних процедур. Після відновлення слід провести всебічну перевірку цілісності систем, застосувати критичні патчі, переглянути й скинути скомпрометовані облікові записи та посилити конфігурації й політики доступу. На підставі результатів розслідування потрібно оновити playbook-и реагування, правила детекції й coverage-matrix по MITRE ATT&CK, а також запланувати додаткові тести емулявання (red/purple-team) для верифікації закриття прогалів. Важливо також прозоро інформувати зацікавлені сторони та, за потреби, зовнішні органи (CERT/регулятори), узгодити остаточне повернення сервісів із бізнес-пріоритетами і запланувати періодичні перевірки для підтвердження стабільності середовища.

Постінцидентний аналіз і навчання — завершальний, але не менш важливий етап: команда проводить after-action review, документує timeline, root-cause, оцінює ефективність застосованих контрзаходів, підраховує збитки й витрати, оновлює IR-playbook і політики безпеки, а також розгортає додаткові технічні й організаційні бар'єри (сегментацію мережі, політики найменших привілеїв, тренінги персоналу, посилений контроль ланцюга постачання ПЗ). Метрики успішності — TTD (time to detect), TTC (time to contain), TTR (time to recover) і кількість вражених хостів — допомагають оцінити прогрес і планувати поліпшення.

Такий системний підхід, що поєднує оперативну ізоляцію, форензичну чіткість, кореляцію розвідданих і продумані процедури відновлення, мінімізує наслідки атак-вимагачів і підвищує стійкість організації до майбутніх загроз. У підсумку, ефективна протидія вимагача потребує одночасного закриття векторів входу і шляхів закріплення, постійного моніторингу поведінки систем і прозорих процедур реагування та відновлення, що робить атаки економічно й технічно менш привабливими для зловмисників.

Крім технічних заходів, необхідно регулярно проводити table-top та purple-team вправи і безпосереднє тестування playbook-ів з подальшим post-incident review, результати якого мають напряду коригувати процедури, сигнатури й політики. Інвестиції в превентивні механізми — мережеву сегментацію, багатофакторну аутентифікацію (MFA), надійні офлайн-резервні копії з контрольованим доступом та шифрування — у поєднанні з постійним поведінковим моніторингом суттєво зменшують можливості для закріплення й екфільтрації даних. Впровадження чіткої політики інформування стейкхолдерів і взаємодії з регуляторами та спільнотами обміну розвідданими (CERT/ISAC), разом із вимірюванням KPI (coverage %, median TTD/TTC, MTTR), забезпечує прозорість, підзвітність і економічне обґрунтування подальших інвестицій у безпеку.

2.2. ЛАНЦЮГ КІБЕРНАПАДУ (CYBER KILL CHAIN) ТА MITRE ATT&CK ЯК ІНСТРУМЕНТИ АНАЛІЗУ ЗАГРОЗ

Ланцюг кібератаки (Cyber Kill Chain) і фреймворк MITRE ATT&CK взаємодоповнюють один одного і разом утворюють практично орієнтований підхід до аналізу загроз. Kill Chain дає лінійну, поетапну модель життєвого циклу атаки — від розвідки до досягнення цілей — що зручно для побудови загального timeline (хронології) інциденту й визначення ключових точок втручання. MITRE ATT&CK (бібліотека MITRE ATT&CK: тактики, техніки й процедури — англ. Tactics, Techniques and Procedures) натомість пропонує деталізований каталог тактик і технік, які супротивники використовують на кожному з етапів; об'єднавши обидва підходи, аналітик отримує і хронологію подій, і конкретні методи для пошуку у телеметрії та побудови детекцій.

Практично це означає проведення мапінгу технік ATT&CK на наявні джерела телеметрії (EDR, Sysmon, DNS, NetFlow, cloud logs) і створення coverage-matrix, яка візуалізує «темні плями» моніторингу. На її основі інженери пріоритизують розробку детекцій, автоматизованих SOAR-playbook-ів і інвестиції в додаткові сенсори там, де ризик для критичних бізнес-сервісів найвищий. Під час інциденту це дозволяє переходити від загальних сповіщень до цільової діагностики — наприклад, швидко визначити, чи використовували Pass-the-Hash, PsExec або

DNS-тунелювання — і миттєво запускати відпрацьовані процедури реагування. Регулярний цикл тестування (емуляція технік, red/purple-team вправи) і відстеження KPI (coverage %, median TTD/TTC, кількість false positives) гарантує, що підхід залишається адаптивним і знижує час від виявлення до нейтралізації загрози.

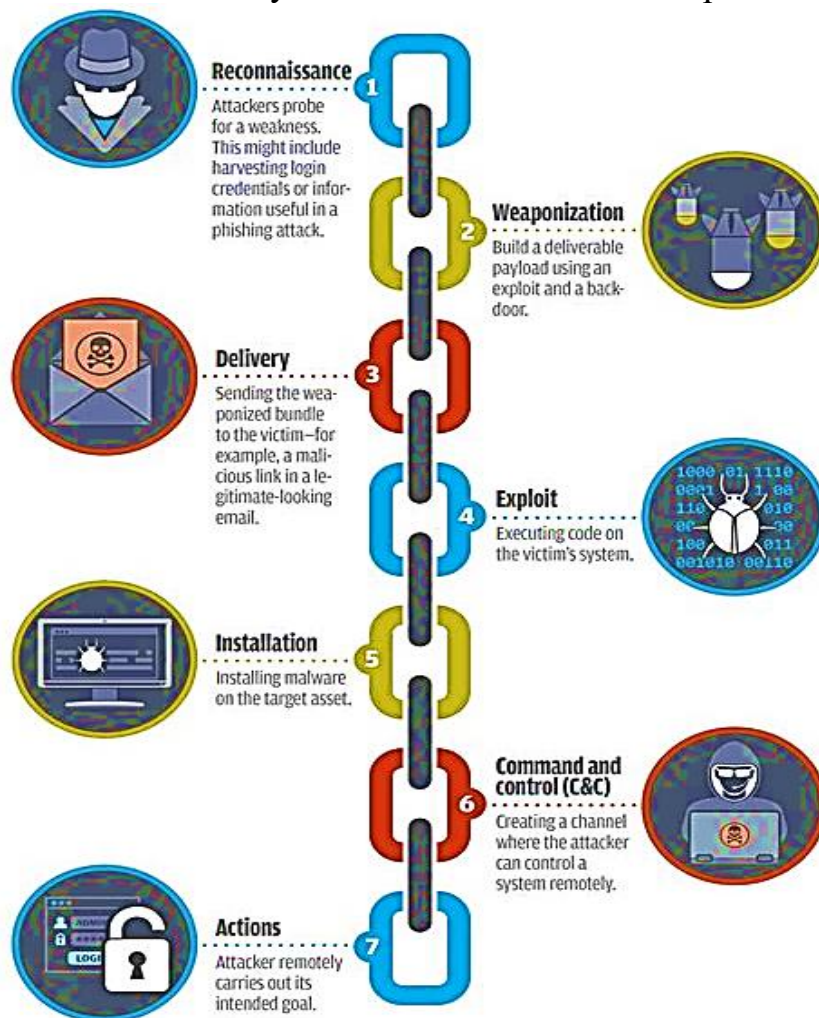


Рис. 2.11. Ланцюг кібератаки (Cyber Kill Chain) за моделлю Lockheed Martin)
Джерело: [1-20, 25-39, 41-45]

Ланцюг кібератаки (Cyber Kill Chain) за моделлю Lockheed Martin — це поетапна модель, яка розбиває цілеспрямовану атаку на повторювані фази з метою полегшити виявлення, переривання й реагування. Ідея полягає в тому, щоб уявити напад як послідовність логічно пов'язаних кроків — від розвідки до досягнення кінцевої мети — і для кожного кроку визначити типові індикатори компрометації (Indicator of Compromise, IOC), можливі техніки та практичні заходи захисту. Такий підхід дає змогу не просто реєструвати окремі події, а будувати цілісну хронологію інциденту (timeline) і переводити розвіддані й спостереження в операційні playbook-и для Центру операцій безпеки (Security Operations Center, SOC).

Перший етап — Reconnaissance (розвідка), під час якого атакувальник збирає публічну й приховану інформацію про ціль: піддомени, відкриті сервіси, технологічний стек, працівників та їхні публічні профілі. Це може бути пасивний збір через WHOIS (сервіс реєстрації доменів) і пошукові запити, або активний скан портів і сервісів. Для захисту важливо обмежити інформаційну видимість (наприклад, приховати контактні дані, знизити публічність профілів співробітників), налаштувати веб-екрани (Web Application Firewall, WAF) і виявлення сканувань через системи виявлення/запобігання вторгненням (Intrusion Detection/Prevention System, IDS/IPS) та кореляцію з threat intelligence; проте слід пам'ятати, що розвідка часто дає перші підказки про мотив і підхід супротивника, тож саме тут закладається початковий контекст розслідування.

Другий етап — Weaponization (озброєння), коли зловмисник готує payload (шкідливе навантаження): пакує експлойт, створює документ із макросом, обфускує виконуваний файл або формує компрометований інсталяційний пакет. На цьому етапі ефективні заходи — sandbox-аналіз файлів, статичний і динамічний аналіз, застосування YARA-правил (правила для ідентифікації бінарних зразків) і політики обмежень на виконувани вкладки в пошті. Хоча озброєння часто відбувається поза межами периметру організації, сигналізація про підозрілі зразки в репозиторіях або нетипові бінарні активності може дати раннє попередження.

Третій етап — Delivery (доставка) — охоплює способи потрапляння payload до жертви: фішинг з шкідливими вкладеннями або посиланнями, drive-by downloads через скомпрометовані вебсайти, атаки через ланцюжок постачання (supply-chain). Захист на цьому етапі спирається на налаштування поштових шлюзів, перевірку підписів і політик DMARC/DKIM/SPF, фільтрацію URL та проксі-аналіз, навчання користувачів (security awareness training) і сегментацію мережі. Детекція підозрілих доставок реалізується через аналіз заголовків пошти, репутації доменів і поведінки користувача щодо викликів на зовнішні ресурси.

Четвертий етап — Exploitation (експлуатація) — це безпосереднє використання вразливості або механізму (наприклад, запуск макроса чи використання уразливості браузера) для досягнення виконання коду на системі жертви. Тут критичною є наявність агентів для моніторингу процесів (Endpoint Detection and Response, EDR), контролю цілісності файлів і механізмів швидкого патч-менеджменту. Виявлення експлуатацій часто базується на кореляції появи незвичних процесів, підозрілих командних рядків і аномальних викликів системних API.

П'ятий етап — Installation / Persistence (встановлення й персистентність) — передбачає налаштування бекдорів і механізмів автозапуску (наприклад, ключі Run/RunOnce у реєстрі Windows, служби, планувальник завдань). Для виявлення таких змін застосовують моніторинг реєстру, сигнатури створення нових служб і перевірки конфігурацій; для запобігання — обмеження прав на створення сервісів,

централізоване управління політиками і контроль за змінами конфігурації. Саме на цьому етапі часто виявляються типові ІОС — нові записи автозапуску, незвичні бінарні модулі або драйвери ядра.

Шостий етап — Command and Control (C2, командування й контроль) — описує налаштування каналу зв'язку між інфікованими хостами і контролюючою інфраструктурою супротивника; це може бути звичайний HTTP(S)-беконінг, DNS-тунелювання або використання легітимних сервісів (наприклад, хмарних платформ, месенджерів) як каналу управління. Виявлення C2 опирається на аналіз DNS-патернів, TLS-fingerprinting (ідентифікація характеристик TLS-з'єднань), аналіз мережеских потоків (NetFlow/IPFIX) і правила Network Detection and Response (NDR). Ефективний захист включає egress-фільтрацію (блокування небажаних вихідних з'єднань), інспекцію SSL/TLS там, де це дозволено, і сегментацію, яка обмежує потенціал C2.

Останній етап — Actions on Objectives (дії за цілями) — охоплює збір даних (Collection), екфільтрацію (Exfiltration) і вплив (Impact), до яких відносяться кража конфіденційної інформації, шифрування даних (ransomware) або саботаж. Захист на цьому етапі опирається на рішення DLP (Data Loss Prevention), аналіз мережевого трафіку (Network Traffic Analysis) і кореляцію подій у системі керування інформацією і подіями безпеки (Security Information and Event Management, SIEM). Практичні заходи включають шифрування даних у спокої, застосування контролю доступу на основі ролей (Role-Based Access Control, RBAC), регулярні резервні копії і політики мінімізації прав.

Щоб модель Kill Chain була операційно корисною, її слід «розгорнути» через фреймворк MITRE ATT&CK (Tactics, Techniques and Procedures — тактики, техніки й процедури): кожен етап ланцюга відповідає набору тактик і технік, які можна картографувати і моніторити. Це дає змогу побудувати coverage-matrix (матрицю покриття), у якій поєднуються джерела телеметрії (EDR, мережеві логи, DNS-журнали, проксі-логи, журнали хмари), конкретні техніки ATT&CK і наявні/відсутні детекції. Така матриця виявляє «темні плями» — техніки, які не покриваються наявною телеметрією — і служить дорожньою картою для інвестицій у додаткові сенсори й інженерні зусилля.

Detection engineering (інженерія виявлення) і threat-hunting (полювання на загрози) — невід'ємні операційні практики в цьому підході: інженери перетворюють ТТР-гіпотези в конкретні правила (наприклад, Kusto Query Language, Elastic Query DSL або універсальний формат Sigma), тестують їх на історичних логах і в середовищі емуляції, а hunt-команди документують сценарії (мета, запити, очікувані артефакти і наступні кроки). Emulation (емуляція) і purple-team вправи (спільні вправи red team і blue team) використовують бібліотеку ATT&CK для моделювання реальних кампаній; red team емулює техніки, blue team

вимірює Time To Detect (TTD — час виявлення) і Time To Contain (TTC — час локалізації), а результати трансформуються в roadmap для підвищення покриття.

Автоматизація реагування через SOAR (Security Orchestration, Automation and Response) дозволяє закріпити операційні playbook-и на конкретні техніки: при виявленні IOC, характерних для Credential Dumping, SOAR може автоматично ізолювати халепний ендпоінт, ініціювати збір memory dump, заблокувати підозрілі облікові записи та застосувати правила мережевого фаєрволу — скорочуючи MTTR (Mean Time To Recovery — середній час відновлення) і зменшуючи людські помилки. Водночас варто наголосити на обмеженнях: модель Kill Chain є лінійною й не відображає рекурсивності або паралелізму реальних атак, а ATT&CK, як описовий ресурс, ефективний лише за наявності якісної, багатоджерельної телеметрії; без агентів EDR і повного мережевого логування багато технік залишаються невидимими.

Отже, практична рекомендація полягає в послідовному підході: спочатку провести інвентаризацію джерел телеметрії й побудувати coverage-map, потім пріоритизувати техніки ATT&CK за ризиком для критичних бізнес-сервісів і частотою застосування в реальних кампаніях, інвестувати в недостаючі сенсори (наприклад, розширене DNS-логування, TLS-fingerprinting, kernel-level telemetry, cloud audit trails), і вже після цього масштабувати розробку правил, SOAR-playbook-ів і регулярні red/purple-team випробування. Вимірювання KPI (Key Performance Indicators — ключові показники ефективності) таких як coverage % (відсоток покриття технік), median TTD, median TTC, кількість false positives на правило та MTTR допоможе економічно обґрунтувати інвестиції й відстежувати прогрес у підвищенні стійкості до загроз.

MITRE ATT&CK (Adversarial Tactics, Techniques, and Common Knowledge — тактики, техніки й процедурні відомості про супротивників) — це каталог поведінкових показників нападників, організований за тактиками (великі цілі атак) і техніками (конкретні методи їх досягнення), який служить практичним інструментом для аналізу загроз, побудови детекцій і організації захисних операцій. У центрі ATT&CK лежить ідея, що ефективне виявлення й реагування повинні базуватися не на статичних сигнатурах, а на поведінкових ознаках (TTP — Tactics, Techniques and Procedures; тактики, техніки й процедури), які повторюються в реальних атаках; це дозволяє переходити від поверхневого пошуку IOC (Indicator of Compromise — індикатор компрометації) до системного розуміння методів супротивника.



Рис. 2.12. Матриця MITRE ATT&CK для корпоративного середовища (Enterprise) з 14 тактиками атак

Джерело: [1-20, 25-39, 41-45]

ATT&CK зручно використовувати як універсальну «мову» між підрозділами: threat intelligence (розвідка загроз) формулює спостереження у вигляді технік і артефактів, інженери з виявлення (detection engineering) трансформують їх у правила для SIEM (Security Information and Event Management — система керування інформацією і подіями безпеки), EDR (Endpoint Detection and Response — рішення для виявлення і реагування на кінцевих точках) і NDR (Network Detection and Response — рішення для виявлення і реагування в мережі), а SOC (Security Operations Center — центр операцій безпеки) використовує ці правила в щоденній роботі. У результаті ATT&CK перетворює розрізнені індикатори на взаємопов'язані ланцюги дій, що описують поведінку ворога від початку проникнення до досягнення цілей.

Практичне застосування ATT&CK починається з мапінгу (mapping): зіставлення наявних джерел телеметрії (журнали Windows Event — журнали подій Windows, Sysmon — System Monitor, мережеві логи, DNS-журнали, cloud audit logs — журнали аудиту хмарних платформ, контейнерні логи тощо) з техніками ATT&CK. Така coverage-matrix (матриця покриття) дає чітку візуалізацію «темних плям» — технік, які невидимі через відсутність відповідної телеметрії — і допомагає пріоритизувати інвестиції: куди додати сенсори, які правила написати в

першу чергу і які playbook-и для SOAR (Security Orchestration, Automation and Response — оркестрація, автоматизація та реагування) підготувати. Без цього кроку АТТ&СК ризикує залишитись вичерпним каталогом без практичної цінності.

Detection engineering (інженерія виявлення) у рамках АТТ&СК має чітку послідовність: отримати гіпотезу з threat intelligence, перетворити її на поведінкове правило або набір правил (наприклад, Kusto Query Language, Elastic Query DSL або універсальний формат Sigma), відтестувати на історичних і емуляційних даних, налаштувати пороги і контекстні фільтри для зниження false positive (хибних спрацьовувань), і лише потім інтегрувати в продуктивну систему моніторингу. Важливо документувати assumptions (припущення) і джерела даних для кожної детекції, щоби потім можна було коректно інтерпретувати сповіщення і швидко проводити розслідування.

Threat hunting (полювання на загрози) на базі АТТ&СК будується як набір гіпотез-ланцюжків: дослідник не чекає ІОС, а ставить питання «як би ворог досяг мети, враховуючи нашу архітектуру і вразливості?», і формує послідовні запити, що перекривають кілька технік. Наприклад, гіпотеза про отримання доступу через фішинг (Initial Access) веде до перевірки запусків нестандартних процесів (Execution), подальшого створення автозапусків (Persistence) і спроб витягти облікові дані (Credential Access). Hunt-ланцюжки мають бути інтегровані в playbook-и з чіткими тригерами, артефактами для збору і кроками ескалації.

АТТ&СК також є основою для емульованих атак і purple-team (кооперативних red/blue) вправ: red team (команда, що емулює напад) обирає набір технік АТТ&СК, які відповідають реальному threat model (моделі загроз) організації, і проводить сценарій, тоді як blue team (захисники) вимірює Time To Detect (TTD — час виявлення), Time To Contain (TTC — час локалізації/утримання) і оцінює якість телеметрії та детекцій. Отримані дані перетворюються на roadmap технічних і процесних робіт: додати сенсори, переписати правила, змінити політики мінімальних привілеїв або оновити SOAR-процедури. Цей цикл (емуляція → вимірювання → поліпшення) є ключовим для підвищення оперативної зрілості (security maturity).

Незамінною частиною інтеграції АТТ&СК у операції є enrichment (збагачення) threat intelligence: автоматичне підключення стрічок ІОС і TTP від постачальників та обмін у секторальних спільнотах (ISAC — Information Sharing and Analysis Center; CERT — Computer Emergency Response Team). Таким чином нові кампанії і техніки швидше потрапляють у внутрішні правила і hunting-сценарії. Водночас важливо фільтрувати і контекстуалізувати ці дані, щоб не завантажувати аналітиків нерелевантними сигналами.

АТТ&СК має і свої обмеження, які потрібно чітко розуміти: по-перше, це описовий, а не нормативний ресурс — він не дає гарантій детекції, якщо відсутня відповідна телеметрія; по-друге, багато сучасних атак використовують living-off-

the-land (використання легітимних інструментів системи), що ускладнює побудову точних правил і підвищує кількість false positive; по-третє, сам обсяг технік і їхніх підкатегорій ускладнює пріоритизацію без чіткої оцінки ризику для бізнес-сервісів. Тому робоча порада — спочатку провести інвентаризацію і створити coverage-matrix, потім пріоритизувати техніки за ризиком і частотою їх застосування проти вашого сектора, і лише потім масштабувати розробку детекцій та автоматизацію.

Для управління ефективністю інтеграції АТТ&СК корисно вводити КРІ (Key Performance Indicators — ключові показники ефективності): відсоток покриття технік (coverage %), median TTD, median TTC, кількість false positives на правило, MTTR (Mean Time To Recovery — середній час відновлення) і кількість успішно пройдених purple-team вправ. Ці метрики дозволяють економічно обґрунтувати інвестиції в додаткові сенсори (EDR-агенти з процесною телеметрією, розширене DNS-логування, TLS-fingerprinting, kernel-level telemetry, cloud audit trails) і відстежувати прогрес.

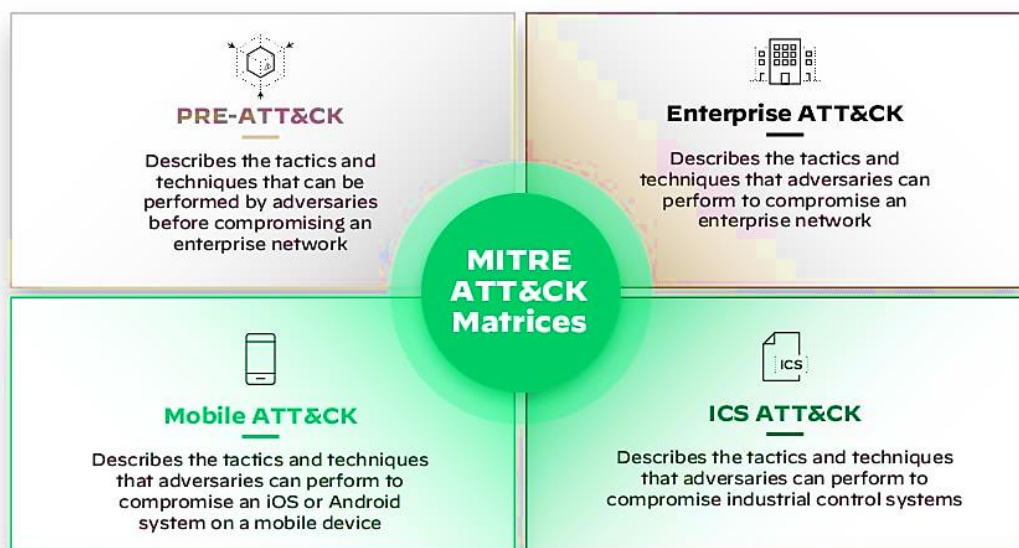


Рис. 2.13. Основні матриці MITRE ATT&CK: PRE-ATT&CK, Enterprise ATT&CK, Mobile ATT&CK та ICS ATT&CK)

Джерело: [1-20, 25-39, 41-45]

Отже, MITRE ATT&CK — це не просто класифікатор: це операційно-орієнтований інструмент, що поєднує threat intelligence, detection engineering, threat hunting і емульовані вправи в єдину екосистему підвищення готовності до атак. Його користь реалізується через практичні кроки — мапінг телеметрії, пріоритизацію технік, написання та тестування правил, регулярні емітації і вимірювання показників — що в результаті дає SOC змогу швидше виявляти, точніше реагувати і системно зменшувати «темні плями» в моніторингу безпеки.

Етапи kill chain — розвідка, підготовка/озброєння (weaponization), доставка (delivery), експлуатація (exploitation), встановлення персистентності

(installation/persistence), командування та контроль (Command and Control, скор. C2) і дії за цілями (екфільтрація даних або вплив на цілі — exfiltration/impact) — можна «розгорнути» у набір технік MITRE ATT&CK: Initial Access (початковий доступ), Execution (виконання), Persistence (персистентність), Privilege Escalation (підвищення привілеїв), Defense Evasion (уникнення захисту), Credential Access (доступ до облікових даних), Discovery (виявлення), Lateral Movement (латеральний рух), Collection (збір даних), Exfiltration (експорт/екфільтрація) тощо. Це перетворює загальну діагностику («виявлено латеральний рух») на конкретні технічні запити: наприклад, чи використовували Pass-the-Hash (передача хеша) або Pass-the-Ticket (передача квитка аутентифікації), чи застосовувався PsExec (утиліта для віддаленого виконання процесів), WMI (Windows Management Instrumentation — інструменти адміністрування Windows) або RDP (Remote Desktop Protocol — Протокол віддаленого робочого столу) — і відповідно формулювати цільові запити у системі керування інформацією і подіями безпеки (Security Information and Event Management, скор. SIEM) та у системі виявлення і реагування на кінцевих точках (Endpoint Detection and Response, скор. EDR).

Практична цінність поєднання полягає в тому, що kill chain допомагає скласти timeline інциденту та виявити етапи, на яких захист провалився, а MITRE ATT&CK деталізує ці провали у вигляді технік, індикаторів компрометації (Indicator of Compromise, скор. IOC) і типової артефактури — це дає можливість оперативно перетворити розвідку та виявлення на повторювані (repeatable) playbook-и для Центру операцій безпеки (Security Operations Center, скор. SOC): шаблони реагування, списки артефактів для судово-експертного аналізу (forensic) і автоматизовані відповіді у системі оркестрації, автоматизації та реагування на інциденти (Security Orchestration, Automation and Response, скор. SOAR). Практично це означає виконання мапінгу технік ATT&CK на конкретні джерела телеметрії (EDR, Sysmon, DNS, NetFlow, cloud logs) і визначення, які артефакти генеруються при кожній техніці. Далі інженери пріоритизують playbook-и за ризиком для критичних бізнес-сервісів і за ймовірністю реалізації техніки в реальних кампаніях, щоб сфокусувати ресурси на найефективніших детекціях. Реалізація включає тестування детекцій через емулявання технік (red/purple-team), автоматичну перевірку правил на історичних логах та впровадження SOAR-процедур з чіткими кроками збору артефактів і containment-дій. Регулярний цикл зворотного зв'язку — моніторинг KPI (coverage %, median TTD/TTC, рівень false positives) і періодичне оновлення карт ATT&CK — забезпечує, що playbook-и залишаються релевантними й реалістично скорочують час від виявлення до нейтралізації інциденту.

CYBER KILL CHAIN VS. MITRE ATT&CK



Рис. 2.14. Порівняння етапів моделі Cyber Kill Chain і тактик фреймворку MITRE ATT&CK

Джерело: [1-20, 25-39, 41-45]

У роботі з телеметрією слід провести мапінг джерел сигналів на техніки MITRE ATT&CK і побудувати coverage-matrix (матрицю покриття) — візуальну карту, які техніки покриваються, а які є «темними плямами». Типові джерела телеметрії: процесні логи кінцевих пристроїв (process start/stop, command line — запуск/зупинка процесів, командний рядок), завантаження модулів, ключі реєстру операційної системи (registry), журнали подій Windows (Windows Event) і Sysmon (System Monitor — агент з пакету Microsoft Sysinternals для розширеного логування), DNS-журнали (Domain Name System logs), проксі/HTTP-журнали (HyperText Transfer Protocol — протокол передачі гіпертексту), мережеві потоки (NetFlow/IPFIX — мережеві флоу-протоколи), TLS-fingerprinting (Transport Layer Security — протокол безпеки транспортного рівня), журнали хмарної платформи (cloud audit logs), журнали середовищ виконання контейнерів (container runtime logs)

тощо. Конкретний приклад: техніку Credential Dumping (витяг облікових даних з пам'яті) логічно шукати у Sysmon (події типу CreateRemoteThread, ProcessAccess), у EDR — підозрілі доступи до процесу LSASS (Local Security Authority Subsystem Service — підсистема локальної служби безпеки), а у мережі — нетипові вихідні передачі або аномальні SMB (Server Message Block) / NTLM (NT LAN Manager) сесії.

Detection engineering (інженерія виявлення) у цьому контексті включає розробку та тестування правил (наприклад, PQL — Pattern Query Language або інша мова запитів платформи, Kusto Query Language — мова запитів Kusto, Elastic Query DSL — мова запитів Elasticsearch, а також Sigma — універсальний формат для правил виявлення), створення поведінкових сигнатур, YARA-правил (інструмент для ідентифікації бінарних зразків) для файлів і бінарів, а також правил для мережевого трафіку (наприклад, нетипові TLS/HTTP-з'єднання, нестандартні user-agent, DGA — Domain Generation Algorithm — алгоритми генерації доменів).

Практичні кроки: 1) для кожної пріоритетної техніки сформулювати гіпотези виявлення; 2) перетворити їх у конкретні правила з контекстними полями (ProcessImage — шлях до виконуваного файлу, CommandLine — командний рядок, ParentProcess — батьківський процес, SrcIP/DstIP — джерело/призначення IP, User — користувач); 3) визначити пороги та контекст для зменшення хибних спрацьовувань (false positives); 4) автоматично тестувати правила на історичних логах і інжектованих емульованих сценаріях. Наприклад, у Kusto Query Language (мова запитів Kusto) простий запит для пошуку підозрілих запусків PowerShell з закодованим рядком може виглядати приблизно так:

DeviceProcessEvents | where FileName == "powershell.exe" and CommandLine contains "-EncodedCommand"

Threat-hunting (полювання на загрози) сценарії повинні базуватися на гіпотезах MITRE ATT&CK — наприклад: «шукати модифікації Run/RunOnce (реєстрові ключі автозапуску), підозрілі служби, невластиві системні виклики, дампи LSASS». Hunt-завдання мають бути документовані: мета, запити, очікувані артефакти, критерії прийняття й наступні дії (investigation playbook — план розслідування). Важливо застосовувати TTP-орієнтовані hunt-chains (ланцюжки пошуку, ґрунтовані на Тактиках, Техніках і Процедурах), а не лише IOC-пошук (пошук індикаторів компрометації).

Emulation (емуляція) та purple-team (спільні вправи червоних і синіх команд) мають використовувати MITRE ATT&CK як бібліотеку технік: red team (чорна або «червона» команда, що емулює нападників) емулює набір технік (з урахуванням локального threat model — моделі загроз), blue team («синя» команда захисту) вимірює Time To Detect (час виявлення, скор. TTD) і Time To Contain (час

локалізації/утримання інциденту, скор. TTC) і закриває прогалини. Результат — roadmap (дорожня карта) для інженерних робіт: які правила треба написати, які сенсори додати, які процедури оновити. Під час вправ корисно фіксувати «прогалини» у coverage-matrix і перетворювати їх у задачі з пріоритетами.

Інтеграція з SOAR (Security Orchestration, Automation and Response — оркестрація, автоматизація та реагування на інциденти) дозволяє прив'язувати автоматизовані playbook-и (сценарії реагування) до конкретних технік MITRE ATT&CK. Приклад: при детекції IOC, пов'язаних із технікою Credential Dumping, SOAR може автоматично ізолювати endpoint (кінцевий пристрій), зібрати дампи пам'яті (memory dump), виконати збір судово-експертних артефактів (process list, registry hives), заблокувати підозрілі облікові записи та розгорнути правило на фаєрволі — усе це скорочує час реагування, знижує людські помилки і забезпечує відтворюваність (repeatability) дій.

Однак кожен підхід має обмеження. Kill Chain — лінійний і може давати хибне відчуття простоти: реальні атаки часто нелінійні, містять паралельні або рекурсивні етапи або повторні входи (re-entry) через backdoor. MITRE ATT&CK — описовий; він корисний лише коли є якісна, багатоджерельна телеметрія. Без EDR-агента (Endpoint Detection and Response — агента для виявлення і реагування на кінцевих точках) з можливістю збирати дані на рівні процесів або без повного мережевого логування багато технік залишаються невидимими (темні плями), а значить — незрозумілими для детекції.

Тому практична рекомендація: починати з оцінки поточної телеметрії і побудови coverage-map (карти покриття), інвентаризації джерел сигналів та визначення пріоритетних «сліпих зон». Інвестувати в джерела сигналів, яких не вистачає (наприклад, розширене DNS-логування, TLS-fingerprinting, instrumented endpoints — кінцеві точки з додатковою інструментацією, kernel-level telemetry — телеметрія на рівні ядра операційної системи, cloud audit trails — журнали аудиту хмари), і лише потім масштабувати правила та автоматизацію. Паралельно варто класифікувати техніки за ризиком для критичних бізнес-сервісів і за частотою використання в реальних кампаніях проти вашої галузі — це допоможе сфокусувати інженерні зусилля.

Ключові показники ефективності (Key Performance Indicators, скор. KPI) та метрики мають бути чітко визначені: відсоток покриття технік MITRE ATT&CK (coverage %), median Time To Detect (медіанний час виявлення), median Time To Contain (медіанний час локалізації/утримання), кількість false positives (хибних спрацювань) на правило, Mean Time To Recovery (середній час відновлення, скор. MTTR) для інцидентів певного класу. Такі метрики допомагають економічно обґрунтувати інвестиції в додаткові датчики чи зміну архітектури і відстежувати прогрес з часом.

Нарешті, потрібно стандартизувати робочі артефакти: шаблони для mapping-звітів (kill chain timeline + MITRE ATT&CK mapping), готові SOAR-playbook-и для технік високого пріоритету, чек-лісти для forensic collection (збір судово-експертних артефактів: memory — пам'ять, disk — диск, network — мережа, registry — реєстр, cloud logs — журнали хмари), а також процес валідації детекцій через регулярні емуляції технік. Регулярне threat intelligence enrichment (підсилення розвідданих про загрози шляхом автоматичного підключення індикаторів компрометації і TTPs від професійних провайдерів), чергові red/purple-team вправи і програма періодичного оновлення карти MITRE ATT&CK допомагають зберігати актуальність. У регіонах із високою загрозою доцільно інтегрувати sector-specific (галузеві) рекомендації і кейси від CERT (Computer Emergency Response Team — команда реагування на інциденти) / ISAC (Information Sharing and Analysis Center — центр обміну та аналізу інформації).

Такий практичний, інтегрований підхід переводить теоретичні моделі у конкретні операційні інструменти, що дозволяє Центру операцій безпеки (Security Operations Center, SOC) швидше виявляти, точніше реагувати і проактивно закривати телеметричні й процесні прогалини — від побудови timeline інциденту до автоматизованих playbook-ів і вимірюваних KPI.

2.3. ТЕХНІКИ УХИЛЕННЯ ТА ПРИХОВУВАННЯ АТАК (ОБФУСКАЦІЯ, ПОЛІМОРФІЗМ, БЕЗФАЙЛОВІ АТАКИ)

Сучасні супротивники дедалі частіше застосовують комплексні методи ухилення та приховування, що роблять класичні сигнатурні підходи неефективними; серед них вирізняються три ключові техніки — обфускація, поліморфізм і безфайлові (fileless) атаки. Кожна з цих технік по-своєму ускладнює статичний аналіз, маскує дійсні наміри й дозволяє зловмиснику залишатися в системі довше, натомість їхня комбінація значно підвищує живучість кампаній. Саме тому сучасна оборона має будуватися на поведінкових індикаторах, багатоваріантних сенсорах (процесна телеметрія, пам'ять, мережеві логи) і систематичних hunt-сценаріях, що дозволяють виявляти не самі файли, а поведінку, яка вказує на злочинні дії.

Обфускація — це набір методів, спрямованих на ускладнення розуміння або аналізу шкідливого коду та його артефактів. Зловмисники обфускують імена функцій, структури бінарів, конкатенують або шифрують рядки, ускладнюють статичний аналіз, використовують поліморфні упаковувачі та шифрування ресурсів.

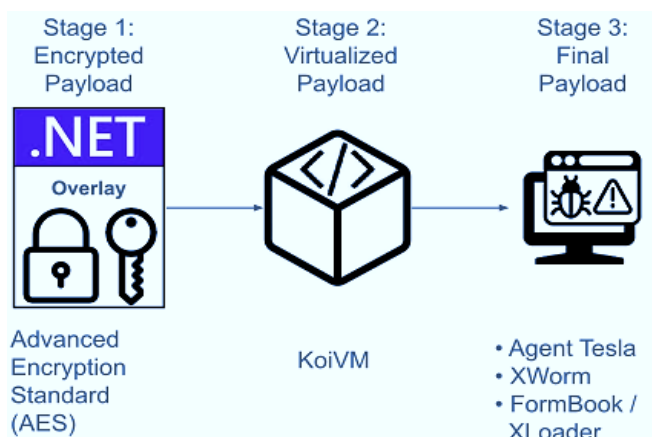


Рис. 2.15. Ланцюг методів обфускації та відповідні техніки протидії (EDR visibility, sandbox-аналіз, memory-forensics, поведінкові сигнатури, ML-моделі)
Джерело: [1-20, 25-39, 41-45, 89-98, 101-120]

Ланцюг методів обфускації — це послідовність взаємопов'язаних прийомів, які атакувальник накладає один на одного навколо свого payload-у з метою ускладнити його виявлення, аналіз і знешкодження; цей «ланцюг» працює як багатошарова оболонка: кожна ланка додає новий рівень приховування, а сумарний ефект значно перевищує простий набір окремих технік.

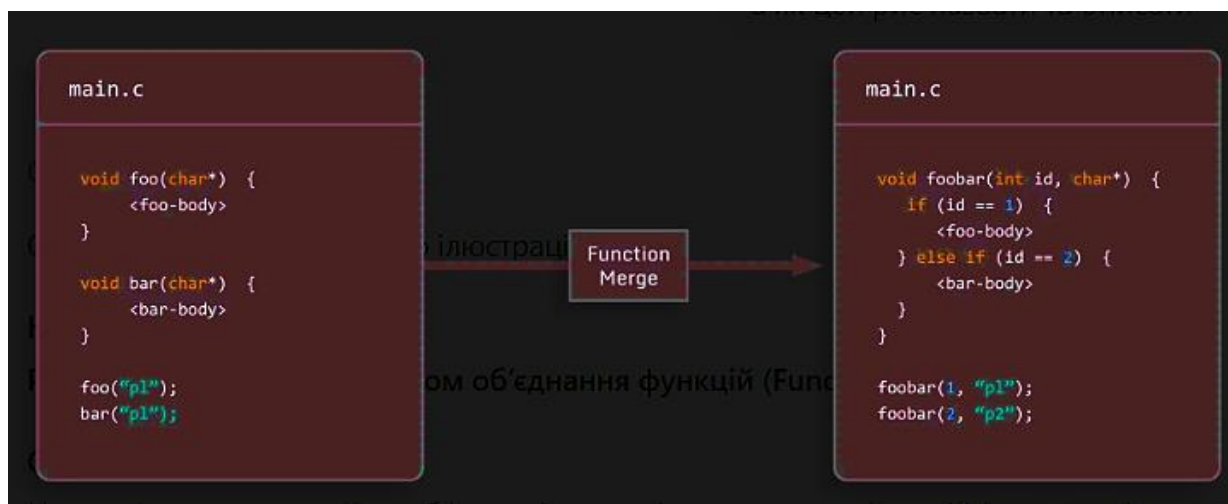


Рис. 2.16. Обфускація шляхом об'єднання функцій (Function Merge)
Джерело: [1-20, 25-39, 41-45, 89-98, 101-119]

На схемі показано прийом обфускації, коли кілька окремих функцій (foo, bar) зливають у одну універсальну функцію (foobar) із «диспетчерським» параметром (id), що всередині гілкує виконання. Така трансформація руйнує початковий граф викликів, ускладнює ідентифікацію меж функцій і розуміння логіки програми під час реверс-інжинірингу. Для статичного аналізу це зменшує кількість впізнаваних

сигнатур і плутає контроль потоку (control-flow), а для динамічного — приховує справжні наміри коду за умовними розгалуженнями. Виявляють подібні трансформації через реконструкцію CFG, пошук атипових «диспетчерів» (великі switch/if із константними селекторами), зіставлення повторно використовуваних блоків коду та динамічний трейсинг шляхів виконання. Як протидія: поведінковий аналіз (EDR), семантичне виділення логічних підфункцій у sandbox/trace, а також автоматичні правила в SIEM/SOAR, що фіксують нетипові гілки викликів після однакових вхідних подій.

Розуміння ланцюга важливе, бо захист має не просто реагувати на окремі артефакти, а розпізнавати рушійну послідовність дій — від простого кодування рядків до складних VM-інтерпретаторів і багатоступеневих in-memory loader-ів — і оркеструвати телеметрію й процеси розслідування відповідно до цієї послідовності.

Початковий шар ланцюга — string obfuscation та basic encoding. На цьому рівні приховуються текстові константи (URL, домени, ключі) за допомогою Base64, простого XOR, custom-encoding або фрагментації рядків, щоб статичний аналіз не показав очевидних індикаторів. Телеметрія при цьому: короткі операції декодування в пам'яті, звернення до функцій декодування, довгі аргументи командного рядка з немовби «шумом». Детекція: правила, що виявляють підозрілі параметри запуску, ScriptBlock/ModuleLogging для скриптових середовищ і прості sandbox-перевірки.

Другий шар — packing та ресурсне шифрування. Payload зберігається як запакований/зашифрований ресурс, а при старті невеликий stub/розпаковувач (decryptor) розшифровує і виконує код у пам'яті. Телеметрія: масивні виділення пам'яті, інтенсивні операції запису у виділені області, виконання коду з нетипових адрес. Для SOC пріоритет — виявлення патернів розпаковування (наприклад, VirtualAlloc → WriteProcessMemory → VirtualProtect → CreateThread) і збирання дамів пам'яті до/після розпаковування, аби витягти розшифрований код.

Третій шар — control-flow obfuscation (заплутування потоку керування). Тут атакувальник міняє структуру виконання: flattening контролю, opaque predicates, вставка «сміттєвих» гілок та часті переходи, що ламають CFG (control-flow graph). Це ускладнює дисасемблінг і автоматичний реверс. Телеметрично це дає фрагментарні послідовності викликів, велике число коротких функцій і непродуктивні API-виклики; детекція — поведінкові сигнатури, що відстежують аномалії в ланцюжках викликів та незвичну тривалість/частоту функцій.

Четвертий шар — instruction substitution, register renaming, junk insertion. Заміна інструкцій на еквівалентні послідовності, перейменування регістрів і додавання непотрібних інструкцій руйнують байтові шаблони. Це підриває прості сигнатури й вимагає переходу до поведінкових фіч: API-послідовностей, файлових операцій, системних викликів. Телеметрія тут — підвищена ентропія коду, зміни у PE-структурі, аномальна частота системних викликів.

П'ятий шар — поліморфізм, де mutation-engine автоматично генерує нові варіанти decryptor-а або упаковки для кожного екземпляра, зберігаючи payload. Це породжує тисячі унікальних файлів з одного сімейства, роблячи хеш-та байтові підписи марними. Для SOC важливо видобувати «сімейні» підписи на основі поведінки (типові послідовності API, патерни розпаковування, таймінги beaconing-у) і класифікувати sandbox-репорти за поведінкою.

Шостий шар — метаморфізм, де сам payload переписується (інструкційні перетворення, перестановки блоків), що ускладнює навіть поведінкову кореляцію. Хоч метаморфізм трапляється рідше, у поєднанні з попередніми шарами він робить автоматичне виявлення надзвичайно складним і вимагає семантичного аналізу та складних ML-моделей.

Сьомий шар — anti-analysis і anti-sandbox техніки: перевірки на debugger/VM, таймінгові пастки (delays), виявлення характеристик sandbox-середовища, шифрування telemetry-виходів. Атакувальники прагнуть змусити зразок «затихнути» під аналізом. Контрзаходи SOC: використовувати розмаїття sandbox-платформ, робити рандомізацію тестових оточень, записувати довші часові інтервали і робити live memory captures.

Восьмий шар — VM/bytecode obfuscation: трансформація логіки в байт-код для власної віртуальної машини, де інструкції інтерпретуються в пам'яті. Це дає дуже високий рівень приховування: детекція потребує витягу внутрішніх таблиць інструкцій і аналізу повторюваних шаблонів інтерпретатора.

Фінальні ланки — multi-stage loaders і reflective/in-memory loading (fileless execution). Багатоступеневі завантажувачі та reflective DLL injection дозволяють обходити диск та маскувати payload під легітимні програми (LOLBin). Телеметрія: ін'єкції в процеси, несподіване створення потоків, швидке поєднання процесної й мережевої аномалії; тут EDR з пам'яттєвим моніторингом і NDR з TLS/HTTP-аналізом критично важливі.

Методика виявлення і протидії ланцюгам обфускації

Методика виявлення і протидії ланцюгам обфускації будується на поєднанні гіпотеза-орієнтованого threat-hunting, автоматизованого детектингу та тісної інтеграції телеметрії з різних шарів інфраструктури. Практичний процес включає формулювання робочих гіпотез (наприклад, «після розпакування у пам'яті з'являються мережеві callback-и на випадкові домени»), відбір і гарантоване збирання джерел (EDR-події процесів і API, Sysmon, PowerShell ScriptBlock/ModuleLogging, NDR/Zeek/Suricata, DNS/Proxy/NetFlow, sandbox-трейси, CloudTrail), виконання кореляцій і пошукових запитів у SIEM та їхню валідацію через memory-forensics і sandbox. Ключові інструменти для реалізації цієї методики — EDR з visibility у пам'ять (з тригерами для автоматичного зняття memory-dump), NDR/IDS для виявлення мережевих патернів і PCAP-записів, SIEM для масштабної агрегації та кореляції, sandbox-платформи для динамічного

виконання й отримання API-послідовностей, інструменти memory-forensics (Volatility/Rekall) для відновлення розпакованого коду, TIP/MISP для збагачення IoC, YARA + поведінкові теги для класифікації репортів та SOAR для автоматизованих плейбуків containment-/forensic-збору.

Detection engineering у цьому контексті орієнтується не на байтові підписи, а на «сімейні» поведінкові ознаки: послідовності API, шаблони розпаковування (VirtualAlloc → WriteProcessMemory → VirtualProtect → CreateRemoteThread), нетипові parent→child ланцюги і швидкі мережеві callback-патерни; ці ознаки нормалізують і кластеризують (DBSCAN/HDBSCAN) із sandbox-трейсів, після чого вони переводяться в Sigma/YARA-правила з поведінковими тегами та в ML-фічі. Для операційного SOC-процесу важливо мати готові SOAR-плейбуки, які при тригері автоматично ізолюють вузол, збирають memory-dump, запускають sandbox-аналіз і оновлюють правила у SIEM/TIP — це дозволяє швидко збирати volatile-артефакти, валідувати гіпотези й ітеративно покращувати детекції, зменшуючи dwell time і підвищуючи покриття проти складних багатошарових схем обфускації.

З точки зору практики, ланцюг обфускації диктує підхід захисту: статичні сигнатури — лише перший фільтр; основа — поведінкові детекції, memory-forensics, sandbox-агрегація і кореляція мережі. SOC має збирати ProcessCreate/CommandLine/ImageLoad, API-виклики (VirtualAlloc, WriteProcessMemory, CreateRemoteThread), дампи пам'яті, network callbacks і sandbox-traces; ці дані нормалізувати, кластеризувати і витягувати «сімейні» підписи.

Detection engineering для ланцюга включає: 1) поведінкові правила (Sigma/Kusto) для конкретних ланцюжків дій (розпаковування → ін'єкція → callback), 2) кластеризацію динамічних ознак з sandbox (DBSCAN/HDBSCAN), 3) YARA-метатеги на поведінкові шаблони, 4) ML-фічі з API-послідовностей і таймінгів beaconing. При підозрі — автоматизовано збирати memory dump, sandbox-репорт і мережеві дампи через SOAR-playbook.

Пом'якшення ризиків: застосовувати AppControl/whitelisting (AppLocker/WDAC), вимагати цифрові підписи для критичних бінарів, обмежувати права і сегментувати мережу, увімкнути ScriptBlock/ModuleLogging для PowerShell, розгортати багаторівневі сенсори (EDR+NDR+sandbox) і регулярні red/purple-team вправи для емулювання багатошарових ланцюгів обфускації.

Форензика: пріоритет — збереження оперативної пам'яті і повних часових журналів під час першої підозрілої активності (бо багато артефактів існують лише в RAM). Збережені sandbox-логи (API sequences, файли, мережа) служать джерелом для кластеризації і побудови сімейних підписів.

Насамкінець — пам'ятаймо, що ланцюг обфускації — це динамічна «гонка озброєнь»: він вимагає від захисту постійного циклу: мапінг телеметрії → розробка детекцій → емулювання/тестування → оновлення правил і сенсорів. Лише така

системна, багат шарова стратегія дає SOC шанс системно виявляти і нейтралізувати навіть найскладніші ланцюги обфускації.

З точки зору телеметрії обфускація проявляється як відсутність читабельних рядків у бінарі, зростання аномальної активності при розпаковуванні на ранньому етапі виконання, часті виклики криптографічних/декомпресійних API в процесі завантаження модуля і нетипові послідовності системних викликів. Для детекції корисні підходи поведінкового аналізу: EDR-тригери на підозрілі моделі розпакування в пам'яті, моніторинг завантаження незвичних модулів, контроль неочікуваних child-process/parent-process ланцюжків і аналіз аномалій у командних рядках (command line). Статичні сигнатури (YARA) часто втрачають силу проти сильної обфускації, тому їх слід поєднувати з динамічним аналізом в sandbox і індикаторами поведінки — наприклад, незвична послідовність API-викликів, підозрілі спроби доступу до LSASS, або різка зміна мережевої поведінки після старту процесу.

В якості заходів пом'якшення — суворе обмеження прав виконання для користувачів, політика блокування виконуваних вкладень, впровадження Application Control (whitelisting) та обов'язковий аналіз підозрілих файлів у sandbox з фіксацією артефактів розпаковування; у forensic-процесі важливо зберегти дампи пам'яті й мережеві сліди розпаковування, оскільки саме пам'ять часто містить дешифровані/розпаковані артефакти, недоступні в статичному образі диска.

Захист від обфускації

Захист від обфускації вимагає системного, багаторівневого підходу: статичні сигнатури — лише перший фільтр, а основна ставка робиться на поведінкову телеметрію, динамічний аналіз і кореляцію сигналів між кінцевими точками, мережею і sandbox-оточенням. Обфускація може ховати рядки, змінювати структуру бінарів або упаковувати ресурси, але вона не здатна повністю приховати структурну і поведінкову логіку атаки — і саме ці сліди стають основою сучасних механізмів захисту.

Практична методика захисту починається з гарантованого збору телеметрії: ProcessCreate/ParentProcess/CommandLine, завантаження модулів (ImageLoad), API-виклики пов'язані з виділенням і записом пам'яті (VirtualAlloc, WriteProcessMemory, VirtualProtect, CreateRemoteThread), DNS/HTTP/NetFlow логи, а також детальні треки з sandbox (API-sequence, файлові операції, таймінги). Критично важливо зберігати memory-dumps при підозрі — багато артефактів розпаковуються або дешифруються виключно в RAM і відсутні на диску. Цю телеметрію збирають EDR з visibility у пам'ять, NDR/IDS для мережевого контексту і централізують у SIEM для масштабних кореляцій і hunting-запитів.

Detection engineering проти обфускації переходить від байтових підписів до «сімейних» поведінкових ознак: шаблонів розпаковування (наприклад VirtualAlloc → WriteProcessMemory → CreateRemoteThread), нетипових parent→child-ланцюгів,

швидких network callback-ів після розпаковування, а також повторюваних API-послідовностей із sandbox-репортів. Ці ознаки нормалізують і кластеризують (DBSCAN/HDBSCAN), переводять у правила Sigma/KQL/Elastic DSL і YARA-маркери з поведінковими тегами; паралельно формуються ML-фічі на основі таймінгів beaconing, ентропії коду та послідовностей syscall. Таке поєднання дає змогу виявляти поліморфні та packed-зразки незалежно від їхніх байтових відмінностей.

Динамічний аналіз у sandbox має бути різноманітним: використовують кілька середовищ з різною конфігурацією (різні ОС, налаштування, таймінги) щоб нейтралізувати anti-sandbox техніки; роблять live memory-capture під час виконання і зберігають повні треки API-викликів. Sandbox-репорти — джерело поведінкових шаблонів для кластеризації й створення «сімейних» підписів; важливо також маркувати репорти метаданими для швидкої агрегації й пошуку.

Інструментальний стек для протидії обфускації включає: EDR з memory-visibility і можливістю автоматичного зняття дамів, SIEM для кореляції і hunting-запитів, NDR/IDS для виявлення мережових аномалій і PCAP-збірки, sandbox-платформи для динамічного аналізу, memory-forensics інструменти (Volatility/Rekall) для глибокого розбору дамів, TIP для збагачення IoC, SOAR для автоматизації збору артефактів і containment-процедур, а також BAS для емулявання складних сценаріїв. Ці компоненти мають бути інтегровані так, щоб при тригері SOAR запуслав плейбук: ізоляція вузла → збір memory-dump → запуск sandbox → оновлення правил у SIEM/TIP і створення тикета в ITSM.

Превентивні заходи значно знижують ризик успіху обфускованих кампаній: застосування Application Control / Whitelisting (AppLocker, WDAC), політика вимоги цифрових підписів для критичних бінарів, суворе обмеження привілеїв користувачів та сегментація мережі. Особливо важливі настройки для PowerShell/Script середовищ: увімкнення ScriptBlockLogging, ModuleLogging, AMSI інтеграція і розгортання Constrained Language Mode там, де це прийнятно. Блокування виконуваних вкладень на поштових шлюзах і обов'язкова перевірка підозрілих файлів у sandbox також значно знижують початковий ризик.

Операційно SOC має мати набір готових hunting-сценаріїв, орієнтованих на ланцюги обфускації — наприклад, виявлення послідовностей розпаковування, інжекцій у процеси, швидких мережових callback-ів на невідомі домени — і тестувати їх регулярними red/purple-team вправами або BAS-симуляціями. Верифікація детекцій і зменшення false positives досягається шляхом поступового інкорпорування поведінкових правил у SIEM і порівняння з історичними образами поведінки для конкретного середовища.

Форензика при підозрі на обфускацію повинна пріоритезувати volatile-дані: збір дамів оперативної пам'яті, повних журналів створення процесів і завантаження модулів, мережових дамів та sandbox-репортів з відтвореними API-

послідовностями. Важливим є збереження chain-of-custody і структуроване збереження артефактів для подальшої кластеризації і побудови «сімейних» підписів, які потім стануть основою автоматичних детекторів.

Нарешті, захист від обфускації — це постійний цикл: моніторинг і збір телеметрії → розробка поведінкових детекцій → емулювання і тестування (BAS, red-team) → впровадження і автоматизація через SOAR → перегляд і оновлення. Лише така комплексна, багатшарова стратегія — поєднання превенції, детекції, динамічного аналізу та автоматизованого реагування — дає SOC шанс системно виявляти й нейтралізувати навіть найскладніші ланцюги обфускації.

Поліморфізм — це специфічний тип обфускації, при якому шкідливий код змінює свій бінарний вигляд від екземпляра до екземпляра, зберігаючи при цьому функціональну поведінку. Поліморфні зразки можуть автоматично підмінювати ключові байти, структуру секцій або застосовувати іншу упаковку, що ускладнює створення тривіальних сигнатур за хешами. На практиці це призводить до великої кількості унікальних зразків з одного сімейства і низької кореляції через статичні індикатори. Для SOC ключовим є перехід від хеш-та підпис-орієнтованих методів до поведінкових сигнатур і побудови «сімейних» ознак: шаблонів поведінки процесів, нетипових сценаріїв мережевого зв'язку, характерних таймінгів beaconing-у тощо. Detection engineering проти поліморфізму включає: кластеризацію зразків по поведінкових ознаках у sandbox, використання YARA-тегів не на основі байтів, а на основі поведінкових атрибутів (API sequence, network callbacks), і розвиток індикацій memory-forensics (шукати постійні структури в оперативній пам'яті, патерни викликів).

Методика виявлення поліморфізму

Методика виявлення поліморфізму будується на поєднанні гіпотеза-орієнтованого hunting та автоматизованих детекцій. На практиці SOC формує гіпотези щодо підозрілої поведінки (наприклад, нетипові ланцюги процесів чи швидке створення мережевих callback-ів після запуску документа) і перевіряє їх за допомогою SIEM-запитів та кореляцій. Ключову роль відіграють EDR-системи з видимістю у пам'ять, які фіксують послідовності API-викликів, і NDR/IDS-платформи, що виявляють аномальний трафік. Для поглибленого аналізу застосовуються sandbox-оточення з кластеризацією зразків за поведінковими ознаками та memory-forensics інструменти (Volatility, Rekall), які дозволяють ідентифікувати спільні структури навіть при різних варіаціях коду.

Важливими допоміжними інструментами є YARA-правила з поведінковими тегами, Threat Intelligence Platforms (MISP, commercial feeds) для збагачення IoC та SOAR, який автоматизує реагування й збір артефактів. Такий комплексний підхід дає змогу перетворювати окремі сигнали на системні детекції й поступово підвищувати покриття SOC проти поліморфних кампаній.

Пом'якшення ризиків — багаторівнева детекція (EDR + NDR + sandbox), блокування виконуваних з тимчасових чи сумнівних директорій, обмеження запуску процедур за допомогою контролю політик і використання механізмів обов'язкової валідації цифрових підписів для критичних бінарів; у розслідуванні звертайте увагу на поведінкові «підписи» сімейства, логування параметрів запуску процесів і збереження повної пам'яті для кореляції.

Поліморфізм — це внутрішній механізм мутації шкідливого коду, при якому кожен новий екземпляр бінарного файлу виглядає по-різному на диску, але зберігає ту саму функціональну поведінку; реалізація зазвичай базується на «mutation engine», який генерує змінені варіанти decryptor-стаба або упаковки, залишаючи payload у зашифрованому чи запакованому вигляді. З архітектурної точки зору поліморфні зразки найчастіше складаються з двох логічних частин — малесенького decryptor (stub), що під час виконання розшифровує чи розпаковує payload у пам'яті, і власне payload, що виконує зловмисну логіку; mutation engine модифікує байти decryptor-а, порядок секцій, вставляє «junk»-інструкції, змінює спосіб шифрування чи структуру заголовків PE, щоб унеможливити кореляцію по хешу або простих байтових шаблонах. Через це класичні AV-підходи, що орієнтуються на хеші або статичні підписи, втрачають ефективність — на практиці одна й та сама кампанія породжує сотні або тисячі унікальних файлів, які не співпадають за хешами і лише частково корелюються за байтовими ознаками.

Важливо відрізнити поліморфізм від метаморфізму: якщо поліморфи зазвичай зберігають payload і змінюють форму stub-а чи шифрування, то метаморфи переписують сам код payload-а (інструкційні підстановки, перестановки блоків) так, що статична структура програми істотно змінюється без очевидного розшифровувача. Для SOC це означає, що у випадку поліморфізму все одно можуть існувати повторювані поведінкові патерни — наприклад, аналогічні послідовності системних викликів під час розпаковування — які слід виявляти й використовувати як основу для детекції.

Щоби мати шанс помітити поліморфні сімейства, необхідно збирати багатопарову телеметрію: процесні події (ProcessCreate, ParentProcess, CommandLine, ImageLoaded), події інструментів пам'яті (CreateRemoteThread, WriteProcessMemory, VirtualAlloc/VirtualProtect), записи про завантаження модулів, повні дампи пам'яті процесів, мережеві логи (DNS, HTTP/S, TLS-fingerprints, NetFlow) та звіти з sandbox-існекцій з детальними послідовностями API-викликів, файловими операціями і таймінгами. Особливе значення має збереження образів оперативної пам'яті, оскільки розпакований або розшифрований код часто існує тільки в RAM і може бути відсутній на диску; без memory-артефактів багато сімейств залишаться «невидимими».

Поведінкові індикатори виявлення проти поліморфізму повинні фокусуватися на ланцюжках дій, а не на байтах: типовими «червоними

прапорцями» є послідовності API, пов'язані з розпакуванням і інжекцією (наприклад, `VirtualAlloc` → `WriteProcessMemory` → `CreateRemoteThread`), різке виділення пам'яті з інтенсивними записами у виділену область, нетипові `parent`→`child` процеси (наприклад, Office документ ініціює нестандартний виконуваний файл), швидка поява `network callback`-ів після розпаковування та характерні таймінги `beaconing`-у. У мережевій телеметрії корисно шукати нетипові `user-agent`, змінні домени, ознаки DGA і аномалії TLS, які разом із процесною поведінкою підвищують пріоритет сповіщення.

Detection engineering для поліморфізму включає кілька взаємодоповнювальних напрямів. По-перше, потрібно збирати й нормалізувати динамічні ознаки з `sandbox` і EDR (API-sequence, file/registry calls, network callbacks, таймінги) для подальшої кластеризації екземплярів: алгоритми типу DBSCAN або HDBSCAN дозволяють виділити «сімейства» на основі поведінки, навіть коли байти відрізняються. По-друге, YARA-правила слід доповнювати метаданими і тегами, що описують поведінку (не лише байтові сигнатури), наприклад, маркувати репорти `sandbox`-у тегами виду «behavior:VirtualAlloc_Write_CreateRemoteThread», що полегшує агрегування. По-третє, варто розвивати підходи memory-forensics: автоматизоване витягнення постійних структур із дампів пам'яті, пошук констант чи ключів розшифровки, а також інструменти для порівняння операційних слідів між екземплярами. Поєднання EDR і NDR дає сильну кореляцію — якщо процес одночасно виконує розпаковування у пам'яті і після цього встановлює нетипове з'єднання, це значно підвищує ймовірність злому.

Практичні hunting-сценарії для SOC мають бути орієнтовані на ці поведінкові патерни: наприклад, шукати події, де процес виділяє пам'ять (`VirtualAlloc/VirtualProtect`), потім виконує багато записів у цю область, після чого створює новий потік або інжектує код у інший процес; або корелювати події розпаковування з появою мережових `callback`-ів на невідомі домени. Навіть прості запити по полях EDR, які відфільтровують звичайні біліші прецеденти (`explorer.exe`, `svchost.exe`), і фокусуються на незвичних `combo-patterns`, дають багато цінної інформації та знижують шум. Ключове — документувати припущення (assumptions) кожної детекції і джерела даних, щоб аналітик розумів обмеження і міг швидко проводити валідацію.

З боку превенції і пом'якшення ризиків корисні перевірені заходи: впровадження Application Control / Whitelisting (AppLocker, WDAC) обмежує запуск невідомих бінарів, політика цифрових підписів для критичних виконуваних файлів знижує ризик запуску підроблених екземплярів, суворе обмеження прав користувачів і сегментація мережі зменшують потенціал розповсюдження. Також важливо налаштувати EDR з видимістю до пам'яті та API-викликів, увімкнути ScriptBlockLogging / ModuleLogging для PowerShell, ввести sandboxing для обробки

підозрілих зразків і забезпечити централізований збір мережевих логів і TLS-fingerprints.

Під час інциденту необхідно пріоритетно збирати повні дампи пам'яті підозрілих процесів, повні журнали створення процесів і завантаження модулів, sandbox-репорти та мережеві дампи, забезпечивши коректний chain-of-custody для доказів. Після аналізу слід кластеризувати знайдені екземпляри, витягнути поведінкові підписи сімейства та оновити детекції й SOAR-playbook-и для автоматичного реагування (наприклад, збір memory dump при першій детекції розпаковувача). Післяінцидентний перегляд має включати оновлення coverage-matrix, закриття виявлених «темних плям» і планування повторних емуляваних вправ для перевірки ефективності нових правил.

Нарешті, варто пам'ятати обмеження: поведінкові правила можуть породжувати false positives, особливо у середовищах з високою динамікою легітимних процесів, а ML-моделі потребують якісних тренувальних даних і регулярного пере-навчання через drift поведінки. Поліморфізм часто комбінується з fileless-та LOLBin-методами, що вимагає глибшої контекстуалізації подій і більш складної кореляції на рівні процесів і мережі. Тому оптимальна стратегія — багатошарова, поведінково-орієнтована оборона з інтегрованими EDR, NDR, sandbox, memory-forensics і постійною програмою threat-hunting та верифікації детекцій, яка дозволяє SOC не лише знаходити окремі екземпляри, а й системно виявляти й нейтралізувати поліморфні сімейства.

Захист від поліморфних атак: методи, інструменти та практичні підходи

Поліморфізм є одним із найскладніших викликів для систем захисту, оскільки він зводить нанівець ефективність класичних сигнатурних методів: кожен новий екземпляр шкідливого коду має унікальний бінарний вигляд, хоча функціональна поведінка залишається незмінною. Це призводить до того, що одна й та сама кампанія може породжувати сотні чи тисячі унікальних файлів, які не збігаються за хешами і лише частково подібні на рівні байтів. Захист від таких загроз можливий лише шляхом відмови від статичних індикаторів на користь поведінкових детекцій і багатошарової телеметрії.

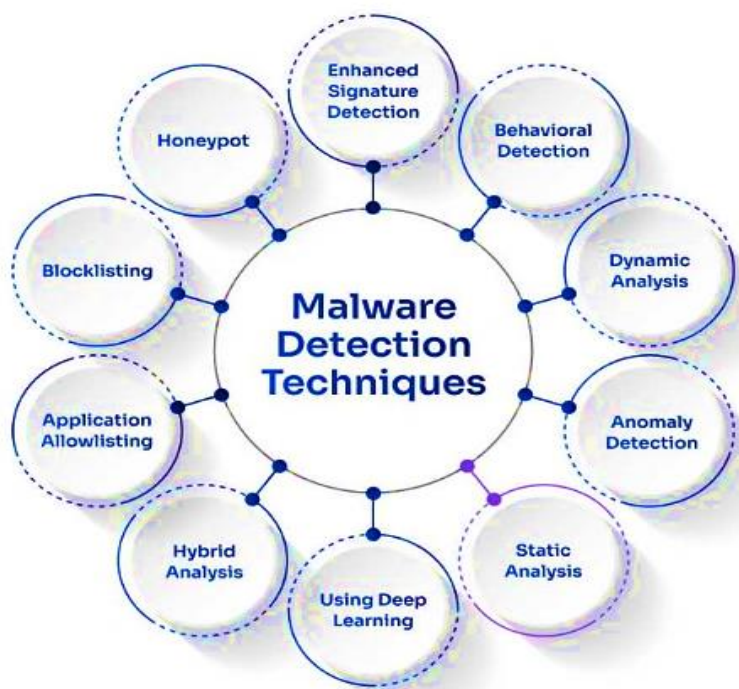


Рис. 2.17. Методи виявлення та захисту від поліморфних атак
Джерело: [1-20, 25-39, 41-45, 89-98, 101-119]

Інфографіка демонструє ключові техніки протидії поліморфному шкідливому ПЗ. Представлені підходи охоплюють як класичні методи (статичний аналіз, підсилене сигнатурне виявлення, блоклістинг), так і сучасні поведінкові та динамічні стратегії (динамічний аналіз, поведінкове виявлення, аномалійний детектинг). Окремо виділено превентивні інструменти (Application Allowlisting), інтегровані методи (гібридний аналіз), а також застосування штучного інтелекту та глибинного навчання для класифікації зразків. Honeypot-технології підтримують збір нових варіантів загроз і збагачують систему знаннями для побудови більш стійких детекцій. Сукупність цих підходів забезпечує багаторівневий захист і дозволяє SOC ефективніше виявляти та нейтралізувати поліморфні сімейства атак.

Методика виявлення і протидії поліморфізму базується на інтеграції кількох ключових напрямів. По-перше, необхідно застосовувати багаторівневу детекцію, поєднуючи EDR із доступом до пам'яті та процесів, NDR для моніторингу мережеских патернів і аномалій, sandbox-платформи для динамічного виконання та memory-forensics для вилучення розпакованого коду, що існує виключно у RAM. По-друге, поведінкові сигнатури замінюють статичні: увага зосереджується на ланцюжках API-викликів (наприклад, VirtualAlloc → WriteProcessMemory → CreateRemoteThread), нетипових ланцюжках parent→child процесів, підозрілих командних рядках і характерних таймінгах beaconing-у.

Додатковим кроком є використання кластеризації та машинного навчання для групування зразків за поведінковими ознаками. Алгоритми на кшталт DBSCAN чи HDBSCAN дозволяють виділити «сімейства» зразків, навіть якщо їхні бінарні форми радикально відрізняються. Для цього широко використовуються sandbox-звіти з API-послідовностями, файловими операціями та мережею. Важливу роль відіграють YARA-правила, збагачені поведінковими тегами, які не просто шукають байтові сигнатури, а відстежують характерні дії.

У практиці SOC превентивні заходи займають центральне місце: впровадження Application Control і Whitelisting (AppLocker, WDAC), вимога цифрових підписів для критично важливих бінарів, суворе обмеження прав користувачів, сегментація мережі та блокування виконання з тимчасових директорій. Це зменшує простір для запуску поліморфних зразків і уповільнює поширення атаки. Форензика й hunting забезпечують глибше розслідування: збір повних дамів пам'яті, логів створення процесів і завантаження модулів, а також кореляція цих даних з мережевими аномаліями.

Ключову роль відіграє автоматизація: SOAR-плейбуки дозволяють швидко ізолювати підозрілий вузол, зібрати volatile-артефакти, відправити зразок у sandbox та оновити правила у SIEM чи TIP. Такий ітеративний процес — від формування гіпотези до оновлення детекцій — скорочує dwell time та знижує ризик непоміченого компромету.

Захист від поліморфізму — це не одноразове рішення, а постійна стратегія, яка базується на поведінковому аналізі, глибокій телеметрії пам'яті та мережі, динамічному аналізі зразків і безперервному вдосконаленні детекцій. Лише комплексний підхід дає можливість не лише знаходити окремі зразки, а й системно виявляти й нейтралізувати цілі поліморфні сімейства.

Безфайлові (fileless) атаки — це підхід, коли шкідлива логіка виконується без запису стійких зловмисних файлів на диск або максимальне використання легітимних інструментів системи (так звані LOLBin — Living Off The Land Binaries) для виконання шкідливих дій. Типові вектори — скрипти PowerShell в пам'яті (часто через -EncodedCommand або через віддалене виконання), зловмисні макроси, ін'єкції в пам'ять інших процесів, reflective DLL injection, використання планувальника або WMI для виконання команд і безпосередня робота з API Windows.

Fileless Attack Techniques

- Phishing / social engineering
- Malicious Office macros
- Malicious Chrome Browser Extension
- Exploit in memory (SMB EternalBlue)
- Supply chain attacks (NotPetya)
- Stolen user credentials

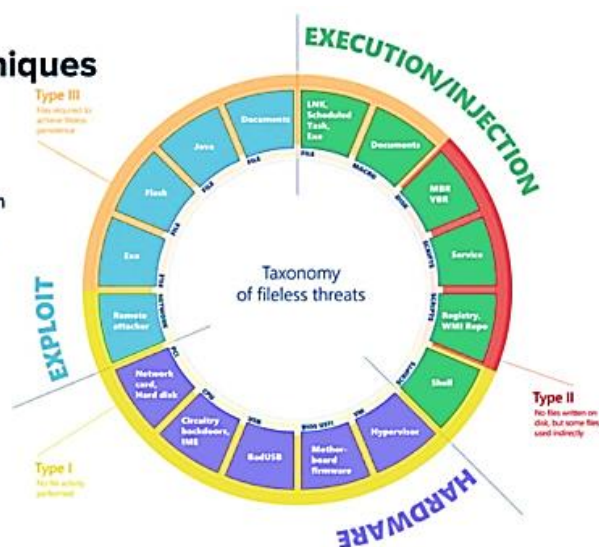


Рис. 2.18. Таксономія безфайлових (fileless) атак — вектори, методи виконання та шляхи приховування
Джерело: [1-20, 25-39, 41-45, 66-72]

Інфографіка ілюструє таксономію безфайлових (fileless) атак: типові вектори початкового доступу (фішинг, шкідливі макроси, компрометації розширень браузера, експлойти в пам'яті, атаки через ланцюг постачання, скомпрометовані облікові дані) і широкий набір механізмів виконання та приховування, які дозволяють зловмисникам діяти без створення помітних стійких артефактів на диску. Головна ідея — шкідлива логіка виконується в оперативній пам'яті або через легітимні інструменти ОС (так звані LOLBins — Living Off The Land Binaries), що робить такі кампанії складними для виявлення традиційними антивірусними підходами. На схемі видно, як початковий exploit/доставка переходить у стадію виконання/інжекції, далі — у механізми персистентності, командування й контроль (C2) і, у кінці, — екфільтрацію або інші цільові дії; навколо цього циклу стоять складні апаратні або прошивочні вектори для найвитонченіших атак.

З технічної точки зору fileless-атаки реалізуються кількома типами методів: виконання скриптів у пам'яті (PowerShell з -EncodedCommand, WMI-скрипти), reflective DLL injection і ін'єкції в інші процеси, запуск команд через rundll32, wmic, schtasks або використання браузерних розширень для виконання коду. Багато сімейств починають із фішингового email-а або зловмисного документу з макросом, який у пам'яті завантажує наступний етап — loader, що розпаковує/дешифрує payload лише в RAM і виконує його без запису на диск. Інші кампанії експлуатують вразливості типу EternalBlue, що дозволяють виконати код у пам'яті віддалено, або крадуть облікові дані, щоб легітимно запуснути команди на віддалених машинах.

Рис. 2.19 ілюструє типовий життєвий цикл безфайлової атаки: на першому етапі зловмисники отримують первинний доступ до цілі через фішинг,

експлуатацію вразливостей або компрометацію облікових даних, після чого запускають код у пам'яті або використовують легітимні інструменти ОС для виконання шкідливих дій; на другому етапі відбувається встановлення персистентності — атакувальник маніпулює реєстром, планувальником завдань, WMI або іншими механізмами, щоб зберегти можливість повторного виконання без створення файлів на диску; нарешті, етап ексфільтрації демонструє, як зібрані дані підготовлюються і передаються назовні через канали C2, приховані FTP/HTTP-з'єднання або інші транспорти. Схема підкреслює послідовність дій і логічний зв'язок між фазами атаки, що має важливі наслідки для захисту: щоб ефективно протидіяти fileless-кампаніям, захист повинен працювати на всіх цих рівнях — запобігання початковому проникненню, раннє виявлення аномальної поведінки в пам'яті та мережі, швидка ізоляція і збір volatile-артефактів для форензики, а також усунення каналів персистентності і блокування ексфільтрації.

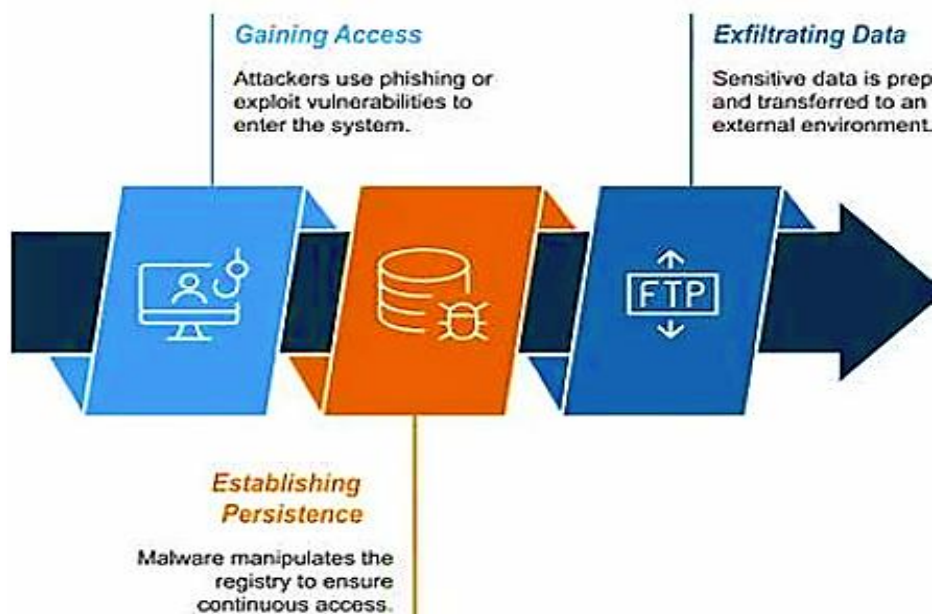


Рис. 2.19. Стадії fileless-атаки (отримання доступу → закріплення → екфільтрація)

Джерело: [1-20, 25-39, 41-45, 68-78, 85-92]

Методика виявлення fileless-активності

Методика виявлення fileless-активності базується на поєднанні гіпотеза-орієнтованого threat hunting та автоматизованих детектів і передбачає чітку послідовність дій, що дозволяє швидко фіксувати volatile-артефакти і переводити знайдені патерни у виробничі детектори. На початковому етапі аналітик формулює конкретну гіпотезу — наприклад, що за останні 24 години на сервері X могли виконуватися PowerShell-скрипти з параметром EncodedCommand або що відбуваються ін'єкції в процеси типу svchost.exe — і прив'язує її до конкретних

ризиків бізнесу. Далі визначається набір джерел телеметрії, необхідних для перевірки гіпотези: події процесів і командні рядки з EDR, записи Sysmon, журнали PowerShell (ScriptBlockLogging, ModuleLogging), мережеві логи (DNS, proxy, NetFlow) та хмарні трейси (CloudTrail), причому важливо гарантувати доступ до сирих даних і достатній retention. На етапі пошуку в SIEM виконуються запити і кореляції (KQL/SPL/Elastic DSL) для виявлення патернів Parent→Child, аномальних cmdline, частих або ентропійно підозрілих DNS-запитів; саме кореляція мережевого та endpoint-сигналу підвищує впевненість виявлення. Якщо запити повертають релевантні підозри, проводиться верифікація через memory-forensics — зняття дамів пам'яті та аналіз цілих образів за допомогою Volatility/rekall — і sandbox-перевірка підозрілих артефактів, щоб підтвердити наявність ін'єкцій або виконаних в пам'яті payload-ів. При підтвердженні інциденту моментально ініціюється containment: вузол ізолюється, блокується C2-інфраструктура (домен/IP) на NGFW/proxy, а SOAR-плейбук автоматизує збір дамів, збагачення через TIP і створення тикетів у ITSM для подальшого forensic-розслідування. На завершальному етапі результати documentуються, IoC і поведінкові патерни додаються у TIP, правила SIEM і плейбуки SOAR оновлюються, після чого відбувається ітеративне тестування через BAS або red/purple-team, аби перетворити знання з однієї hunting-сесії в постійні автоматичні механізми детекції; ключова особливість цієї методики — швидкість фіксації volatile-даних та ітеративне вдосконалення детекторів для скорочення dwell time.

Наслідком відсутності або крихкості дискових артефактів є зміщення фокусу в бік пам'яті та поведінки: ключова телеметрія — це події створення процесів (ProcessCreate), параметри запуску (CommandLine), ланцюги Parent→Child, API-виклики, завантаження модулів (DLLLoad), операції виділення пам'яті (VirtualAlloc/VirtualProtect), інжекції (WriteProcessMemory / CreateRemoteThread) і мережеві callback-патерни (DNS-запити, нетипові HTTP/TLS з'єднання, DGA). Через це EDR-платформи з видимістю в пам'ять, включно з можливістю робити memory-dumps і логувати API-виклики, є критично необхідними для детекції fileless-загроз; паралельно NDR/Network telemetry (TLS-fingerprinting, аналіз DNS та NetFlow) допомагає фіксувати C2 та екфільтрацію.

Для практичного виявлення та реагування рекомендується комплексний набір заходів. По-перше, увімкнути й налаштувати AMSI (Antimalware Scan Interface) і детальний журнал PowerShell (ScriptBlockLogging, ModuleLogging), а також розглянути Constrained Language Mode для обмеження можливостей скриптів. По-друге, впровадити EDR із пам'яттевою видимістю, активними тригерами на типові ланцюжки інжекції (VirtualAlloc→WriteProcessMemory→CreateRemoteThread) та можливістю автоматичного збору memory-dump при підозрі. По-третє, використовувати NDR для виявлення callback-патернів, DGA-діяльності, аномалій TLS та нетипових User-

Agent; кореляція EDR+NDR у SIEM дає значно вищий рівень впевненості при піднятті пріоритету інциденту.

Превентивні заходи не менш важливі: блокування макросів за замовчуванням у поштових клієнтах і офісних пакетах, фільтрація вкладень на gateway, застосування політик MFA для дистанційного доступу, регулярний патч-менеджмент (щоб закрити вразливості, що дозволяють виконувати код у пам'яті), мережева сегментація й обмеження вихідного трафіку (egress filtering). Додатково рекомендовано впровадження Application Control / Whitelisting (AppLocker, WDAC), обмеження запуску з тимчасових директорій і контроль використання LOLBins у критичних середовищах.

Інцидент-реагування при fileless-підозрі має пріоритетно фіксувати volatile-артефакти: робити повні або часткові дампи оперативної пам'яті процесів, збирати ProcessCreate/CommandLine журнали, мережеві дампи і sandbox-репорти. Оскільки розпакований код і ключі часто зникають при перезапуску, швидкість збору є ключовим фактором — SOAR-playbook-и мають автоматично ізолювати хост і ініціювати збір forensic-артефактів при спрацюванні тригерів. Важливо також документувати chain-of-custody і зберігати всі артефакти для подальшого аналізу й можливих юридичних дій.

Операційно SOC має розробити конкретні hunting-сценарії та coverage-matrix: зіставити техніки fileless-атак з доступною телеметрією (EDR process events, memory dumps, Sysmon, DNS logs, proxy logs, NetFlow, cloud audit logs) і виявити «темні плями», де потрібні додаткові сенсори. Регулярні red/purple-team вправи, що емулюють fileless-сценарії (PowerShell-Laden, reflective injection, WMI abuse), повинні перевіряти працездатність детекцій і швидкість реакції. KPI — coverage %, median TTD/TTC, частка інцидентів з успішно зібраним memory-dump — допоможуть відстежувати прогрес і обґрунтовувати інвестиції.

Підсумок: інфографіка нагадує, що fileless-атаки — це не один прийом, а ціла екосистема векторів і технік, що акцентують увагу захисту на пам'яті, поведінці та кореляції сигналів між endpoint і мережею. Ефективний захист вимагає поєднання превенції (фільтрація пошти, патчі, MFA, whitelisting), детекції (EDR з memory-visibility, NDR, AMSI/PowerShell logging), автоматизованого реагування (SOAR) та постійного тестування через емульовані атаки — лише така багатошарова стратегія дає шанс виявити, ізолювати і нейтралізувати fileless-кампанії до того, як вони досягнуть своїх цілей.

Fileless-атаки особливо небезпечні, бо традиційні антивіруси й сигнатурні системи, що орієнтовані на файли, можуть їх пропустити. Для виявлення потрібно фокусуватися на телеметрії поведінки: підозрілі запуски PowerShell з нетиповими аргументами, обхід AMSI (Antimalware Scan Interface) ознак, неочікувана активність wmic чи schtasks, ін'єкції в процеси типу explorer.exe або svchost.exe, підозрілі з'єднання після виконання скрипта. EDR-платформи з мониторингом API-

викликів, інструменти memory-forensics та можливість виконати повний захоп пам'яті при підозрі є критично важливими для розслідування fileless-інцидентів. Пом'якшення включає заборону PowerShell-скриптів з невідомих джерел, політики Constrained Language Mode для PowerShell, моніторинг і контроль використання LOLBins, налаштування AMSI та розширення можливостей інспекції TLS/HTTPS, де це можливе. У forensic-процесі ключове значення мають дампи пам'яті, журнал подій PowerShell (включно з ModuleLogging, ScriptBlockLogging) та мережеві сліди, оскільки диск часто не містить корисної інформації; важливо також зібрати артефакти автентичності запуску (ParentProcess, CommandLine, ProcessCreate events) для відновлення ланцюжка виконання.

Ключові інструменти і приклади їх застосування

Для ефективного threat hunting та швидкого реагування на fileless-атаки необхідно поєднувати кілька спеціалізованих інструментів у єдину, взаємодіючу екосистему, де кожен компонент виконує свою роль і підсилює інші. На рівні кінцевих точок основну базу телеметрії забезпечують EDR-платформи з видимістю в пам'ять (наприклад, CrowdStrike, SentinelOne, Microsoft Defender, Cortex): вони фіксують події ProcessCreate, повні CommandLine, завантаження DLL, трекінг викликів API і можуть автоматично ініціювати збір memory-dump при типовій послідовності інжекції (VirtualAlloc → WriteProcessMemory → CreateRemoteThread). Саме ця рантайм-телеметрія дозволяє виявляти ін'єкції, reflective DLL-завантаження і виконання скриптів у пам'яті, які залишають мало або зовсім не залишають дискових артефактів.

Щоб надати контекст і масштаби подіям з кінцевих точок, EDR інтегрується зі SIEM (Splunk, Elastic, QRadar), де відбувається агрегація логів, виконання масових пошукових запитів і кореляційних алгоритмів. У SIEM поєднують мережеві патерни (DNS/Proxy/NetFlow) з подіями EDR, що дає змогу відрізнити локальний «шум» від реальної ланцюгової атаки: наприклад, підозрілий PowerShell-запуск на хості і одночасні beacon-DNS-запити формують високопріоритетний інцидент. SIEM виступає як «мотор» кореляцій і збереження історії, необхідної для hunting-запитів і post-mortem forensic-аналізу.

Аналіз мережевого боку здійснює NDR/IDS-рівень (Zeek, Suricata та комерційні NDR-платформи), який фіксує ознаки DNS-тунелювання, DGA-діяльності, аномалії TLS/HTTP та дозволяє збирати PCAP для детального forensic-дослідження. Поєднання NDR з EDR дає критично важливу кореляцію «хто спілкується з ким»: NDR показує з ким спілкується вузол у мережі, EDR — як цей вузол поведився перед і під час з'єднання. Разом вони дозволяють встановити ланцюг дій від початкового виконання до ексфільтрації.

Для глибокого технічного розслідування volatile-артефактів застосовують memory-forensics інструменти (Volatility, Rekall), які дозволяють відновити ін'єкції, небезпечні модулі в пам'яті, приховані нитки та дампи стеку процесів; ці

інструменти часто використовують як наступний крок після автоматичного захоплення пам'яті EDR-агентом. Паралельно з інструментами пам'яті слід активувати можливості платформи Windows — ScriptBlockLogging і ModuleLogging для PowerShell та AMSI (Antimalware Scan Interface), які дають змогу інспектувати і логувати виконувані скрипти в рантаймі й підвищують шанси виявлення обфускованих payload-ів, навіть якщо вони виконуються в пам'яті.

Щоб прискорити пріоритизацію та підвищити точність, кожну знахідку варто збагачувати даними з TIP (MISP або комерційні фіди): перевірка доменів, IP-адрес, хешів файлів і відповідних TTP дозволяє швидко визначити ступінь ризику і вирішити, чи підлягає подія негайному containment. Саме на підставі цього збагачення SOAR-платформи (Cortex XSOAR, Splunk SOAR, IBM Resilient) виконують автоматизовані плейбуки: ізоляцію вузла, блокування доменів/адрес на NGFW, ініціювання memory-dump, створення тикетів у ITSM і повідомлення аналітиків — усе це робиться швидко і повторювано, зменшуючи MTTD/MTTR.

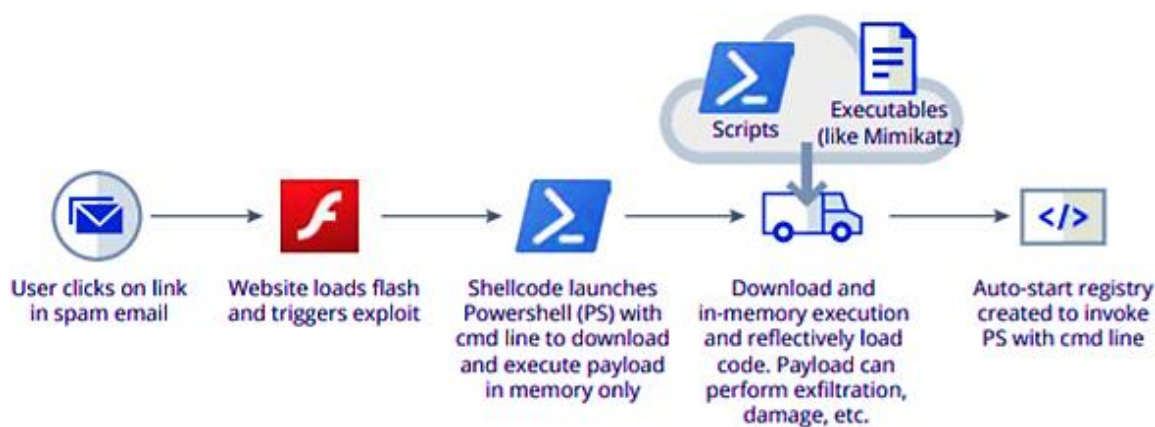


Рис. 2.17. Приклад ланцюга знищень безфайлової атаки
Джерело: [1-20, 25-39, 41-45]

Нарешті, для перевірки й постійного удосконалення покриття детекторів використовують BAS та red/purple-team інструменти (AttackIQ, SafeBreach), які емулюють fileless-TTP і дозволяють оцінити, чи виявляється емульована атака існуючими правилами і наскільки ефективні playbook-и. У сукупності ці інструменти формують збалансований стек: EDR дає відповіді «що саме виконувалося», NDR — «з ким воно спілкувалося», SIEM — «як поєднати події в інцидент», SOAR — «як автоматично і повторювано відреагувати», memory-forensics і AMSI — «як підтвердити in-memory активність», а TIP і BAS — «як пріоритизувати і тестувати». Тільки така скоординована архітектура дає змогу SOC ефективно виявляти, збирати докази і нейтралізувати fileless-кампанії до того, як вони нанесуть критичну шкоду.

Усі три техніки взаємопов'язані: поліморфізм і обфускація підвищують ефективність fileless-підходів, а fileless-диспетчеризація використовує ОС-засоби, щоб приховати сліди на диску. Отже, захист має бути багат шаровим і орієнтованим на поведінку — поєднання EDR (аналіз процесів і пам'яті), NDR (аналіз мережевого трафіку), SIEM-кореляцій, sandbox-аналізу і threat-intelligence enrichment. Практично SOC має: 1) побудувати coverage-matrix для цих типів телеметрії, 2) розробити спеціальні hunt-сценарії (наприклад, пошук atypical PowerShell encoded команд у логах, підозрілих модулів, частих create/terminate циклів процесів), 3) впровадити SOAR-playbook-и для швидкої ізоляції й збору memory dumps, і 4) регулярно виконувати емулявані атаки (red/purple team) з метою валідації детекцій і закриття «темних плям».

Захист від безфайлових (fileless) атак

Захист від безфайлових (fileless) атак вимагає комплексного підходу, оскільки традиційні антивірусні рішення, що орієнтуються на файли, тут неефективні. Основний акцент робиться на поведінковій аналітиці та моніторингу пам'яті: EDR-рішення повинні вміти відстежувати API-виклики (VirtualAlloc, WriteProcessMemory, CreateRemoteThread), а також формувати ланцюги Parent→Child процесів, щоб виявляти інжекції та нетипові сценарії виконання. Додатково застосовуються NDR та мережеві сенсори для аналізу DNS-запитів, TLS-fingerprinting і пошуку DGA-патернів, що допомагають фіксувати канали C2 та екфільтрації.

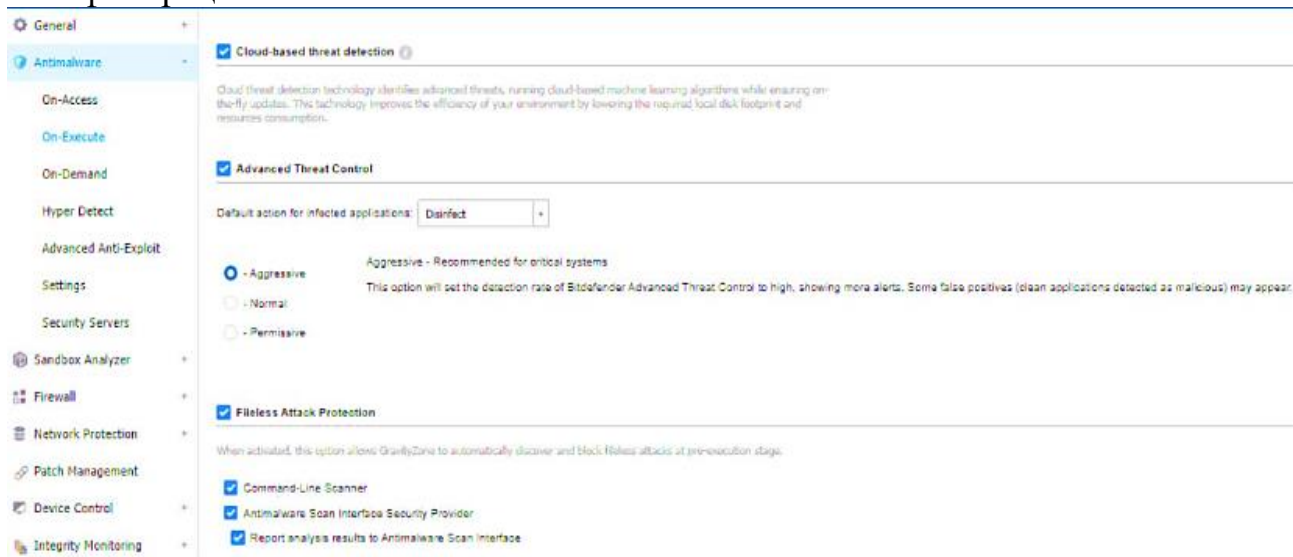


Рис. 2.20. Механізми захисту від безфайлових (fileless) атак

Джерело: [1-20, 25-39, 41-45, 68-78, 85-92]

Рис. 2.20 демонструє ключові технічні підходи, які використовуються сучасними системами безпеки для протидії безфайловим загрозам. Основний акцент зроблено на багаторівневій перевірці та блокуванні шкідливих дій ще до їх виконання. На схемі показано, що захист включає аналіз командних рядків для

виявлення підозрілих параметрів запуску, використання AMSI (Antimalware Scan Interface) для інспекції скриптів у реальному часі, блокування небезпечних сценаріїв PowerShell, а також перевірку пам'яті перед виконанням інжекцій чи завантаженням шкідливого коду. Завдяки цим механізмам система здатна ідентифікувати й зупиняти атаки, які обходять класичні антивірусні рішення, що орієнтовані виключно на файли. Такий підхід підкреслює значення поведінкового аналізу та контролю операцій у пам'яті як базових інструментів сучасного захисту від fileless-загроз.

Превентивні заходи включають: блокування макросів за замовчуванням, застосування багатофакторної автентифікації, обмеження прав користувачів, регулярний патч-менеджмент, контроль запуску з тимчасових директорій і використання Application Whitelisting (AppLocker, WDAC). Важливо увімкнути AMSI та детальний аудит PowerShell (ScriptBlockLogging, ModuleLogging), а також використовувати Constrained Language Mode для зниження ризиків.

SOAR-платформи забезпечують швидке реагування: автоматичну ізоляцію вузла, збирання дамів пам'яті та журналів, запуск sandbox-аналізу, блокування C2-адрес у реальному часі. У forensic-аналізі пріоритет мають volatile-артефакти: пам'ять процесів, журнали ProcessCreate/CommandLine, мережеві дампи, оскільки саме вони несуть докази атаки.

Нарешті, захист від fileless-загроз вимагає постійного удосконалення детекцій: формування поведінкових сигнатур, оновлення IoC у TIP, впровадження сценаріїв hunting та регулярних red/purple-team вправ для емулявання атак. Така багатопланова стратегія дозволяє знизити dwell time і підвищити ймовірність вчасного виявлення та нейтралізації загроз.

Індикатори компрометації та техніки полювання (hunting)

Навіть при застосуванні приховувальних технік залишаються артефакти поведінки: нетипові мережеві з'єднання (DNS-тунелювання, незвичні шифровані потоки), одноразові великі операції пам'яті, незвично часті виклики системних API, незвичні часи виконання скриптів. Під час threat hunting використовують: побудову профілів нормальної поведінки (UEBA), пошук аномалій у потоках DNS/HTTP, аналіз відмінностей у базових білінгових даних (наприклад, раптові запити на зовнішні IP), кореляцію з TIP. Робочі інструменти — EDR консоль, SIEM-кореляції, YARA, Zeek/Suricata, sandbox.

На рис. 2.21 представлено процес проведення threat hunting, який складається з чотирьох взаємопов'язаних етапів. Спершу формується гіпотеза, що передбачає ідентифікацію підозрілої мережевої активності та аналіз потенційних вразливостей у системах. Далі виконується розслідування, яке включає перегляд журналів для

виявлення нетипових шаблонів доступу та аналіз даних кінцевих точок на наявність аномалій. Наступним кроком є виявлення нових патернів, зокрема блокування зловмисних IP-адрес у реальному часі та оновлення правил міжмережевих екранів з метою запобігання доступу. Завершальний етап охоплює реагування, збагачення та автоматизацію: на основі знайдених артефактів удосконалюються політики безпеки, впроваджується автоматизоване виявлення й реагування на загрози. Зображення також демонструє ключові переваги процесу: підвищення ефективності виявлення загроз на 67 %, скорочення часу реагування на інциденти на 46 % та покращення загальної стійкості безпеки підприємства на 79 %.

Process to perform cyber threat hunting

This slide showcases four staged process to perform cyber threat hunting. It includes stages: hypothesis, investigation, new patterns and respond.

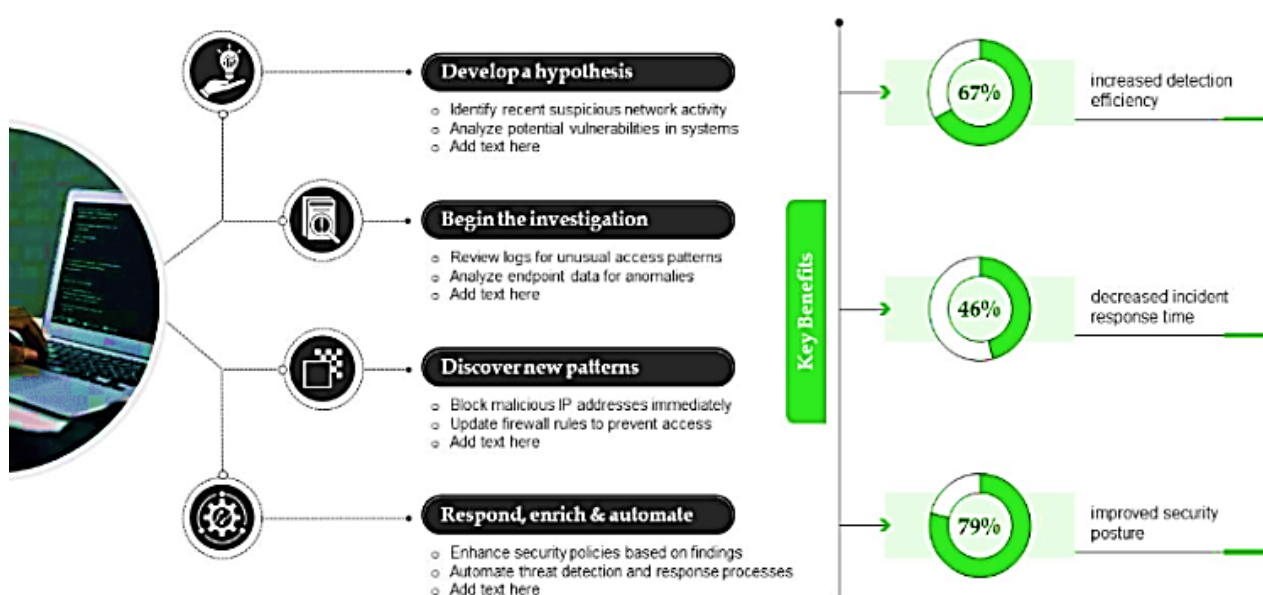


Рис. 2.21. Процес проведення threat hunting

Джерело: [1-20, 25-39, 41-45, 68-78, 85-92]

Індикатори компрометації (IoC) та техніки полювання (threat hunting) — це пов'язані підходи до виявлення та розслідування проникнень: перше — конкретні артефакти або сигнали, що свідчать про компрометацію, друге — проактивний процес пошуку прихованих загроз у середовищі, часто до того, як вони перетворяться на інцидент. Вони тісно взаємодоповнюють одне одного: IoC дають базу для технічної ідентифікації слідів атак, а полювання дозволяє шукати приховані прояви загроз навіть без явних індикаторів. У практиці SOC IoC часто використовуються як «тригери» для формування правил у SIEM, тоді як hunting зосереджується на виявленні нетипових поведінкових сценаріїв. Завдяки цьому організація отримує глибший рівень видимості і може знаходити як відомі, так і невідомі атаки. Важливим аспектом є постійне оновлення бази IoC з платформ Threat Intelligence, що дозволяє швидко реагувати на нові кампанії APT. Полювання

ж, у свою чергу, допомагає виявляти TTP, які ще не мають готових IoC, та формувати нові правила детекції. Такий підхід забезпечує баланс між реактивним і проактивним захистом, підвищуючи кіберстійкість підприємства.

Індикатори компрометації — це будь-які сліди, які дають натяк на злочинну активність: IP-адреси і домени C2, хеші шкідливих файлів, підозрілі рядки в реєстрі, незвичні запити DNS (DNS-тунелювання), аномальні процеси у пам'яті, незвичні права доступу або зміни в конфігурації систем. IoC бувають статичними (хеш файлу), динамічними (поведінкові патерни) і контекстними (час виконання, геолокація, прив'язка до конкретного користувача чи сервісу). Використання індикаторів компрометації є ключовим елементом роботи SOC та систем моніторингу, адже вони дозволяють швидко виявляти та блокувати відомі загрози. Статичні IoC зазвичай застосовуються у сигнатурному аналізі антивірусів і IDS/IPS, тоді як динамічні та контекстні допомагають будувати поведінкові моделі та виявляти нові типи атак. Важливо, що ефективність IoC залежить від їхньої актуальності, оскільки застарілі індикатори швидко втрачають цінність через еволюцію загроз. Для підвищення результативності їх об'єднують із Threat Intelligence-платформами, які постійно оновлюють базу даних IoC у реальному часі. Крім того, IoC часто стають основою для побудови правил у SIEM-системах і сценаріїв реагування в SOAR. Таким чином, грамотне використання індикаторів компрометації дозволяє значно скоротити час виявлення атак і підвищити рівень захищеності інфраструктури.

Threat hunting — це структурований, гіпотеза-орієнтований процес: мислити як нападник, формувати гіпотези (наприклад «хтось з внутрішньої мережі встановив backdoor, використовуючи PowerShell»), збирати дані з різних джерел і перевіряти гіпотези за допомогою запитів, кореляцій і ручного аналізу. Hunting поєднує використання SIEM-запитів, EDR-консолі, мережевого трафіку (Zeek/Suricata), логів хмарних сервісів і даних з TTP. На відміну від класичного реагування на інциденти, яке починається з отриманого алерта, threat hunting ініціюється самими аналітиками з метою виявлення прихованих або малопомітних загроз. Цей процес зазвичай базується на гіпотезах, що формуються з урахуванням MITRE ATT&CK-матриці, відомих TTP (tactics, techniques, procedures) противників і власного досвіду команди SOC. Для перевірки гіпотез застосовуються як автоматизовані аналітичні інструменти, так і ручні методи аналізу, що дозволяє знаходити загрози, які можуть не генерувати сигнатурних попереджень. Успішне полювання завершується не лише виявленням потенційної атаки, а й створенням нових правил кореляції, оновленням плейбуків SOAR та доповненням бази знань. Крім того, результати hunting допомагають удосконалювати профілі поведінкової аналітики UEBA та підвищувати якість індикаторів компрометації. Таким чином, threat hunting є проактивною складовою сучасного захисту, що перетворює SOC із реактивного центру у стратегічний інструмент попередження атак.

Методика полювання за загрозами (threat hunting)

Методика полювання за загрозами (threat hunting) є проактивним підходом до кіберзахисту, який спрямований на виявлення прихованих атак і підозрілої активності ще до того, як вони перетворяться на підтверджений інцидент. На відміну від традиційного реагування, що починається з автоматично згенерованих алертів, threat hunting базується на ініціативі аналітиків безпеки, які виходять із гіпотези про можливу компрометацію й активно шукають докази у логах, телеметрії та мережевому трафіку. Основними принципами цього підходу є гіпотеза-орієнтованість, тобто формування перевірюваних припущень; контекстність, що враховує бізнес-критичність активів і ролі користувачів; ітеративність із постійним удосконаленням правил і плейбуків; а також поєднання автоматизованого аналізу з ручними методами розслідування.



Рис. 2.22. Фреймворк (структура) процесу threat hunting
Джерело: [1-20, 25-39, 41-45, 68-78, 85-92]

На рис. 2.22 представлено фреймворк процесу threat hunting, який відображає циклічну структуру полювання за загрозами. Першим кроком є розширення видимості, що передбачає збір максимально повної телеметрії з кінцевих точок, мережевих сенсорів, систем логування та хмарних сервісів для створення бази даних про події. Далі здійснюється аналіз розвідданих, під час якого використовуються джерела Threat Intelligence для виявлення актуальних тактик, технік і процедур противника. На основі цього формується гіпотеза щодо можливих сценаріїв атак чи підозрілої активності в інфраструктурі. Наступним етапом є

створення аналітики для полювання, що включає формування правил пошуку, кореляційних запитів у SIEM або моделей аналізу поведінки. Після цього безпосередньо виконується процес hunting, під час якого проводиться пошук аномалій, перевірка гіпотез і виявлення прихованих загроз. Отримані результати стають основою для розслідування та подальших дій, які дозволяють підтвердити або спростувати наявність інцидентів, уточнити їхній контекст і визначити рівень ризику. Завершальним кроком є перегляд і вдосконалення аналітики, що забезпечує оновлення правил, моделей та сценаріїв детекції для підвищення ефективності майбутніх сесій hunting. Таким чином, фреймворк демонструє безперервний цикл, у якому кожен етап взаємопов'язаний і спрямований на постійне вдосконалення захисту та підвищення стійкості інфраструктури до сучасних кібератак.

Процес полювання складається з кількох послідовних етапів. Спершу визначається область пошуку (scoring) і формулюється гіпотеза, наприклад: «в інфраструктурі можуть бути сліди fileless-атаки через PowerShell». Далі здійснюється збір телеметрії з різних джерел — систем EDR, SIEM, мережевих сенсорів, хмарних сервісів, IAM, поштових шлюзів. Після цього аналітики створюють пошукові запити, виконують кореляцію подій та застосовують інструменти аналізу пам'яті чи мережевого трафіку. У випадку виявлення підозрілих артефактів проводиться їх верифікація, збагачення контекстом за допомогою Threat Intelligence та, за потреби, ескалація для реагування через SOAR або вручну. Завершальний етап передбачає lessons learned: уточнення правил SIEM, оновлення плейбуків SOAR, створення нових IoC та доповнення бази знань SOC.

Методика полювання спирається на низку ключових інструментів. SIEM використовується для агрегації й пошукових запитів у великих масивах логів, EDR — для аналізу поведінки процесів і пам'яті на кінцевих точках, мережеві системи Zeek чи Suricata — для контролю трафіку. Для збагачення результатів залучаються TTP-платформи, що містять інформацію про кампанії APT, IoC та нові TTP, а для тестування ефективності нових правил застосовуються BAS-рішення. Важливим принципом є документування кожної сесії hunting: від опису гіпотези та запитів до висновків і рекомендацій, що дозволяє повторювати процес і підвищувати рівень зрілості SOC. Таким чином, threat hunting стає не разовою дією, а циклічним процесом, який перетворює реагування на інциденти з реактивного підходу на стратегічне проактивне управління загрозами, зменшуючи час перебування атакера у системі (dwell time) та підвищуючи стійкість усієї інфраструктури.

Типові hunting-сигнатури й індикатори

Типові hunting-сигнатури й індикатори відіграють ключову роль у проактивному пошуку загроз, оскільки вони дозволяють аналітикам SOC виявляти неочевидні або приховані атаки ще до того, як вони переростуть у серйозні інциденти. На відміну від класичних сигнатур IDS/IPS, що реагують виключно на відомі шаблони атак, hunting-сигнатури орієнтовані на аномалії та поведінкові

відхилення, які вказують на можливу злочинну активність. До таких індикаторів належать різноманітні ознаки: від аномальної автентифікації, коли користувачі здійснюють вхід у систему з незвичних геолокацій або поза робочим часом, до підозрілої поведінки процесів, наприклад запуску PowerShell з нетиповими аргументами чи створення скриптів у тимчасових директоріях. Важливим сигналом можуть бути і мережеві ознаки — з'єднання з невідомими IP-адресами, DNS-запити зі випадково згенерованими доменами або тривалі HTTPS-сесії без активності користувача, які часто вказують на тунелювання чи командно-контрольну активність (C2).

Крім цього, увагу привертають файлові артефакти, зокрема створення нових виконуваних файлів без цифрового підпису, поява рідкісних драйверів або модифікація системних каталогів. Не менш значущими є інсайдерські індикатори, серед яких масове копіювання конфіденційних документів, вивантаження великих обсягів інформації у хмарні сервіси чи несанкціоновані зміни прав доступу. Такі ознаки, навіть якщо поодинокі вони не виглядають критичними, у сукупності створюють картину атаки і дозволяють аналітикам перевіряти свої гіпотези. У сучасних умовах hunting-сигнатури формуються як вручну, так і за допомогою автоматизованих систем: UEBA аналізує поведінку користувачів і пристроїв, ML/AI-моделі виявляють приховані закономірності, а Threat Intelligence-платформи забезпечують актуальні IoC у реальному часі.

Застосування таких індикаторів інтегрується в SIEM для побудови правил кореляції, а у SOAR — для запуску автоматизованих плейбуків реагування. Завдяки цьому hunting не обмежується ручним аналізом, а перетворюється на комплексний процес, де слабкі сигнали об'єднуються в цілісні сценарії атак. У результаті організації отримують не лише можливість швидше виявляти нові загрози, а й активно зміцнювати кіберстійкість своєї інфраструктури, переводячи кіберзахист із реактивного у стратегічно проактивний режим.

Інструменти й методи threat hunting

Інструменти й методи threat hunting охоплюють набір технологій, аналітичних підходів і практик, які разом дають змогу виявляти приховані, малопомітні або нові форми атак — від побудови гіпотез до верифікації, ескалації й оновлення детекторів. У центрі процесу — джерела телеметрії: журнали кінцевих точок (EDR), системні логи і аутентифікація (AD, IdP), мережеві логи та сенсори (NetFlow, Zeek, Suricata), DNS-логи, проксі/HTTP-логи, хмарні трейс-логи (CloudTrail/Azure Activity), поштові логи і дані з CMDB/asset-inventory. Без повного, контекстного набору цих даних hunting втрачає ефективність — тому перша і постійна задача аналітика та інженера — забезпечити повноту, коректність та збереження телеметрії (retention).

На рис. 2.23 зображено цикл процесу threat hunting, який відображає логіку побудови та вдосконалення полювання за загрозами. Процес починається з

формування гіпотези (Propose Hypothesis), що ґрунтується на знаннях про можливі атаки, розвідданих та поведінкових відхиленнях у системі. Наступним етапом є збір даних (Acquire Data) з різних джерел — систем моніторингу, журналів подій, телеметрії з кінцевих точок чи мережевих сенсорів. Після цього дані підлягають аналізу (Analyze Data) для виявлення аномалій, підозрілих активностей чи підтвердження гіпотези. Далі результати аналізу узагальнюються та збагачуються контекстом (Inform and Enrich), що дозволяє точніше визначити сутність виявленої активності та сформулювати нові знання для SOC. Отримана інформація використовується для вдосконалення механізмів детекції (Enhance Detections), шляхом оновлення правил у SIEM, плейбуків у SOAR чи моделей поведінкової аналітики. У випадку підтвердження підозри ідентифікується реальна загроза (Valid Threat), яка переходить у подальші етапи реагування. Завдяки цьому процес threat hunting є безперервним і циклічним: кожне нове полювання збагачує базу знань, удосконалює системи виявлення та забезпечує підвищення стійкості інфраструктури до кібератак.

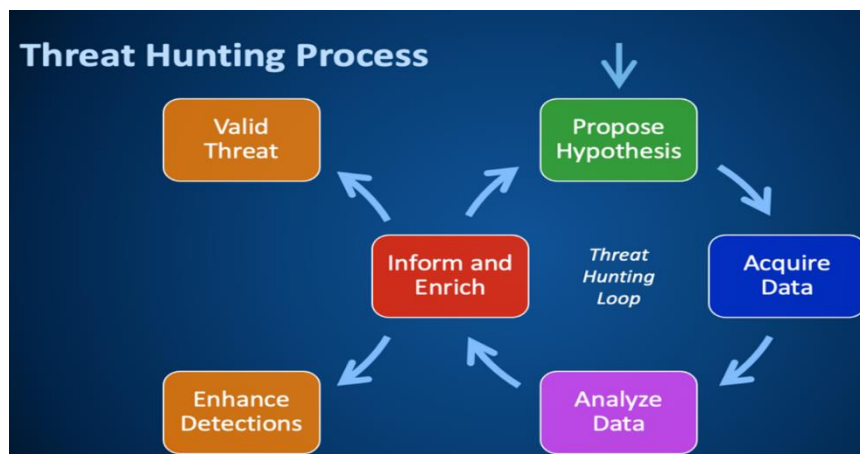


Рис. 2.23. Процес threat hunting (цикл полювання за загрозами)
Джерело: [1-20, 25-39, 41-45, 68-78, 85-92]

Технічно hunting опирається на кілька ключових платформ. SIEM служить для агрегування й збереження логів, виконання масових пошукових запитів і кореляцій; EDR дає детальну рантайм-телеметрію кінцевих точок (процеси, командні рядки, ін'єкції, модулі пам'яті) і можливість швидкої ізоляції; мережеві сенсори (Zeek, Suricata) дозволяють аналізувати трафік, виявляти аномалії DNS/HTTP і збирати PCAP; TIP/Threat Intelligence збагачують знайдені події контекстом (IoC, TTP, фіди APT). Для автоматизації збору контексту й виконання рутинних дій застосовують SOAR, а для перевірки та валідації захисту — BAS/Attack Simulation (SafeBreach, AttackIQ). Інструменти пам'яті та форензики

(Volatility, Rekall), YARA для сканування артефактів і набори для мережевого аналізу доповнюють стек аналітика.

Методи hunting поєднують кілька підходів: гіпотеза-орієнтований (формулюється припущення про TTP і перевіряється набором запитів), індикаторний (пошук по IoC), поведінковий (UEBA — User & Entity Behavior Analytics) і статистичний/аналітичний (використання ML для виявлення аномалій). У практиці це означає: формулювання гіпотези (наприклад, «можливий DNS-tunneling з боку серверу X»), підбір джерел, написання пошукового запиту (KQL, SPL, Elastic DSL, SQL-подібні запити), кореляція результатів, ручний аналіз артефактів і — за потреби — запуск плейбука в SOAR для containment. Для уніфікації правил часто використовують Sigma-формати для опису детекторів та YARA-шаблони для файлів.

Аналітичні техніки включають побудову поведінкових профілів (baseline), виявлення аномалій у часових рядах (time-series analysis), аналіз ланцюгів подій (graph analytics — побудова графів атак з вузлами «користувач», «хост», «IP», «домен»), кореляцію системних подій із мережевою телеметрією, а також memory forensics для підтвердження in-memory активності. Часто застосовують методи enrichment: кожен підозрілий артефакт збагачують даними з TTP, WHOIS, репутації IP/доменів, CMDB (власник, критичність), щоб пріоритизувати знахідки.

Практичні прийоми hunting — це шаблони запитів і playbooks. Наприклад, пошук підозрілих PowerShell-запусків із параметрами EncodedCommand у SIEM у поєднанні з аналізом мережевих з'єднань того ж хоста; пошук нетипових DNS-запитів за ентропією доменного імені; виявлення lateral movement за транзакціями між хостами з одночасними успішними логінами і запуском підозрілих процесів. На додачу, автоматизовані симуляції (BAS) дають змогу перевірити, чи виявить поточна конфігурація захисту певну техніку, і слугують вхідним сигналом для нових hunting-гіпотез.

Організаційно важливі практики: документування кожної hunting-сесії (гіпотеза → джерела → запити → знахідки → дії), багаторівнева пріоритизація по ризику, регулярні purple-team вправи для перевірки детекторів, ітеративне оновлення правил SIEM/SOAR та навчання ML-моделей на реальних інцидентах. Метрики ефективності hunting включають кількість сесій, знайдені релевантні IoC на сесію, скорочення dwell time, поліпшення MTTD/MTTR і долю автоматизованих реагувань.

Нарешті, слід пам'ятати про виклики: великий обсяг даних і шум, потреба в якісній телеметрії та утриманні довгих історій логів, дефіцит кваліфікованих аналітиків. Рекомендації — інвестувати в якість даних (збір, контекст, retention), впроваджувати стандартизовані hunting-шаблони (Sigma/YARA), використовувати SOAR для рутинної автоматизації, поєднувати ручний аналіз з ML-підтримкою та регулярно тестувати систему через BAS/red-team, щоб трансформувати hunting із

реактивної активності в регулярну проактивну практику, що істотно підвищує стійкість організації до кібератак.

Захист від атак, що виявляються за допомогою індикаторів компрометації та технік threat hunting

Захист від атак, що виявляються за допомогою індикаторів компрометації та технік threat hunting, базується на поєднанні реактивних і проактивних засобів. Використання IoC дозволяє оперативно блокувати відомі загрози за рахунок застосування сигнатур у антивірусах і IDS/IPS, створення правил у SIEM для виявлення IP-адрес C2-серверів, хешів шкідливих файлів чи підозрілих DNS-запитів, а також інтеграції контекстних індикаторів у системи UEBA для аналізу поведінкових відхилень.

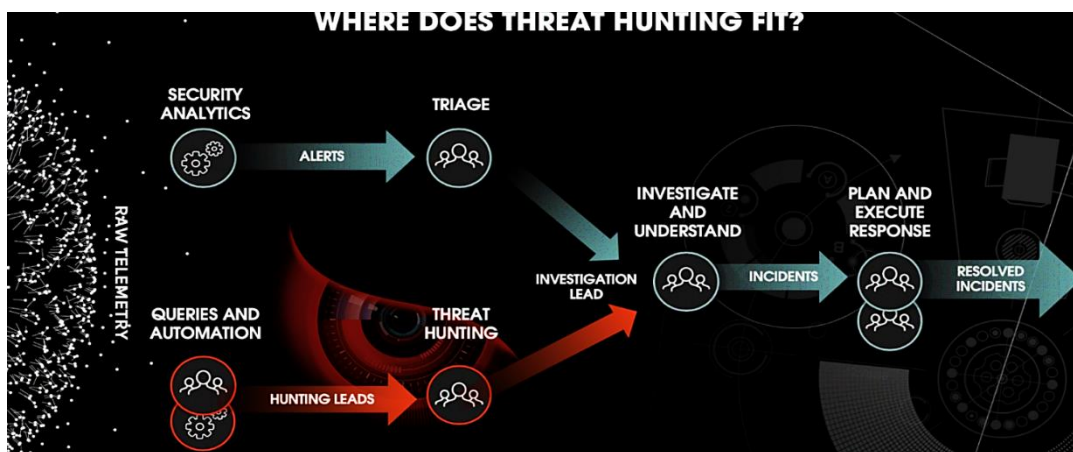


Рис. 2.24. Місце threat hunting у процесі виявлення та реагування на інциденти
Джерело: [1-20, 25-39, 41-45, 68-78, 85-92]

На рис. 2.24 представлено схему інтеграції threat hunting у загальний цикл кіберзахисту. Вихідною точкою є необроблена телеметрія (raw telemetry), що надходить із систем моніторингу та кінцевих точок. Частина цих даних опрацьовується засобами аналітики безпеки (security analytics), які формують автоматичні сповіщення та передають їх на етап тріажу, де визначається пріоритетність інцидентів і відбираються події для подальшого розслідування. Інша частина телеметрії використовується у процесах автоматизації та спеціальних запитів, які стають основою для проведення threat hunting. У ході полювання аналітики формують гіпотези, виконують пошук прихованих ознак атак та генерують hunting leads, що перетворюються на нові напрямки розслідувань (investigation leads). Далі як автоматично згенеровані алерти, так і результати hunting сходяться у спільному етапі «investigate and understand», де загрози аналізуються та класифікуються. Наступним кроком є планування та виконання заходів реагування, що дозволяє перевести інциденти у категорію «resolved incidents». Таким чином, схема демонструє, що threat hunting доповнює класичну модель SOC, збагачуючи

процес виявлення та реагування додатковими сигналами і забезпечуючи виявлення прихованих атак, які не були б ідентифіковані лише стандартною аналітикою безпеки.

Ефективність IoC значною мірою залежить від актуальності, тому їх постійне оновлення через Threat Intelligence-платформи є ключовим чинником швидкого реагування на нові кампанії APT. Водночас threat hunting забезпечує пошук прихованих загроз, які ще не мають готових IoC, і передбачає гіпотеза-орієнтований підхід: аналітики SOC формують припущення про можливу атаку, перевіряють їх за допомогою кореляції логів, аналізу мережевого трафіку, даних з EDR та хмарних сервісів, а також використовують інструменти на кшталт Zeek, Suricata, YARA чи memory forensics. Важливо, що результати полювання не лише допомагають виявити приховані TTP, а й збагачують базу IoC, удосконалюють правила SIEM і плейбуки SOAR, що підвищує рівень зрілості SOC.

Організаційно захист підсилюється регулярним документуванням сесій hunting, проведенням purple team вправ для перевірки детекторів, навчанням ML-моделей на реальних інцидентах та автоматизацією рутинних дій через SOAR. У підсумку поєднання актуальних індикаторів компрометації та систематичного threat hunting забезпечує баланс між реактивним і проактивним підходом до безпеки, знижує dwell time, скорочує час реагування на інциденти та формує стійку інфраструктуру, здатну протидіяти як відомим, так і новим загрозам.

2.4. МЕТОДИ СОЦІАЛЬНОЇ ІНЖЕНЕРІЇ ТА ФІШИНГОВІ КАМПАНІЇ

Соціальна інженерія та фішингові кампанії — це системні атаки на людський фактор і довіру, які поєднують ретельну розвідку, психологічні тригери і технічні канали доставки, щоб змусити користувача розкрити облікові дані, виконати шкідливу дію або завантажити зловмисне ПЗ. Вони не є випадковими «спам»-повідомленнями: сучасні кампанії часто таргетовані (spear-phishing, whaling), багатоступеневі і можуть включати голосові дзвінки (vishing), SMS (smishing), атаки через соціальні мережі, а також використовувати компрометовані вебсайти й месенджери. У центрі успіху лежить розуміння організаційної структури, ролей людей і їхніх процесів — тому захист має бути одночасно технічним і поведінковим.

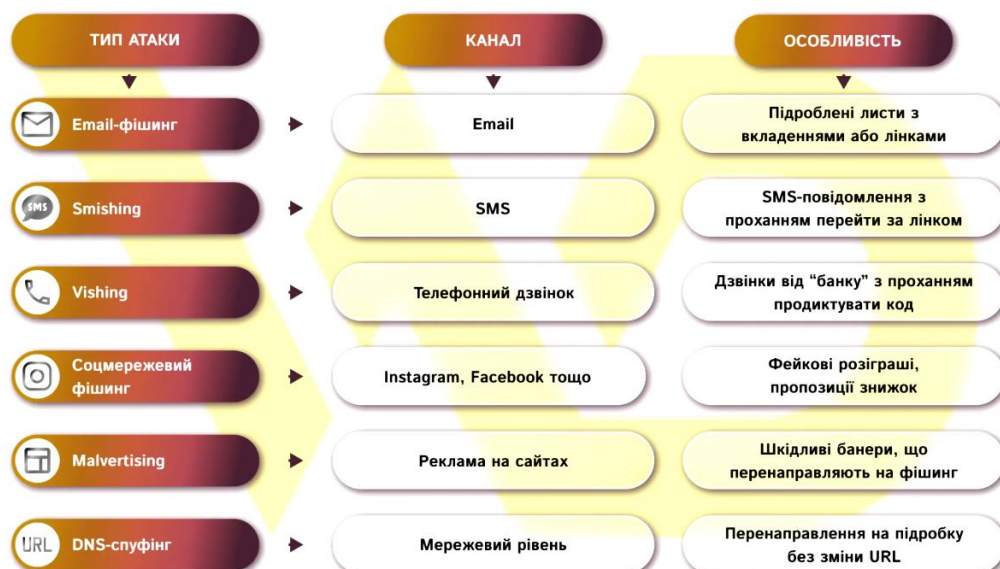


Рис. 2.25. Види фішингу та їх відмінності
Джерело: [1-20, 25-39, 41-45]



Рис. 2.26. Стислий перелік ключових кроків, щоб у разі фішингової загрози діяти швидко та послідовно
Джерело: [1-20, 25-39, 41-45]

Перша фаза — розвідка (reconnaissance). Атакувальники збирають публічну й приховану інформацію: LinkedIn-профілі, організаційні діаграми, корпоративні сайти, публічні документи, записи DNS і WHOIS, а також можливі

витоки паролів. Ці дані дозволяють створити реалістичні pretext-сценарії (виправдання звернення), ідентифікувати впливових осіб у компанії (ціль «whale») і підібрати часові рамки або місцеві обставини для більшої правдоподібності. Часто розвідка включає й аналіз технічних векторів: чи використовує організація певного постачальника, чи є відкриті VPN/портالي, які можна використати як лінки у фішинговому листі.

Атакувальники часто автоматизують збір OSINT-даних за допомогою скриптів і сервісів (переліку email-адрес, піддоменів, публічних профілів співробітників) та використовують інтернет-скани для виявлення відкритих сервісів і вразливих точок доступу. Вони корелюють інформацію з «витоків» і баз даних зламаних облікових записів (paste-сайти, репозиторії витоків), щоб знайти повторно використані паролі або підтверджені email-адреси цілей. Соціальні мережі, пости в блогах і публічні календарі переглядають для виявлення подорожей, відпусток або внутрішніх подій, які можна використати як правдоподібний контекст (наприклад, термінова перевірка рахунків під час відсутності керівника). Часто зловмисники реєструють «схожі» домени (typosquatting), готують фейкові сторінки входу або налаштовують інфраструктуру відправки пошти, щоб підвищити рівень довіри і доставляння фішингових повідомлень. Перед масовою кампанією проводяться пробні відправки й A/B-тести приманок, щоб відсортувати слабкі варіанти і відточити формулювання, теми та час розсилки. У підсумку якісна розвідка поєднує технічні і людські дані, даючи змогу створити багатошарові, персоналізовані pretext-сценарії, які значно підвищують ймовірність успіху атаки.

Другий етап — підготовка приманки (lure crafting) — базується на психології: авторитет (листи начебто від керівника або відомої організації), терміновість/страх (негайно оновіть дані), привабливість (фінансова можливість), соціальне підтвердження (інші робили так само) або цікавість (корисний документ). Хороші фішингові повідомлення працюють на емоціях та контексті — наприклад, «звіт за Q2 для підпису» або «оновлення пароля перед мережею офлайн у вихідні». Важливо відзначити: описуючи ці прийоми в навчальних цілях, ми підкреслюємо, як захищатися, але у жодному разі не надаємо шаблонів чи кроків, які полегшують створення шкідливих матеріалів.

Атакувальники зазвичай максимально персоналізують тему та зміст листа, підставляючи імена, посади або деталі про поточні проєкти, щоб підвищити правдоподібність і знизити пильність отримувача. Вони часто використовують елементи соціального доказу — згадки колег, посилання на реальні внутрішні документи або події — щоб створити відчуття законності звернення і посилити психологічний тиск. Перед масштабною розсилкою зловмисники можуть проводити тестові A/B-відправки й невеликі пробні кампанії, щоб виміряти open-/click-rate і відшліфувати формулювання, час відправлення та формат приманки. У

відповідь на це захист має впроваджувати багаторівневі технічні бар'єри: валідацію заголовків, суворе застосування SPF/DKIM/DMARC, реальне-часове сканування URL та sandbox-детонацію вкладень. Не менш важливими є процесні контрзаходи — стандартизовані процедури перевірки фінансових чи кадрових запитів (двоетапне погодження, перевірка по відомих каналах) та регулярні навчання співробітників із практичними прикладами соціальної інженерії. Швидка реакція SOC — кореляція кліків, моніторинг незвичних логінів і автоматичні тригери для захисту (наприклад, примусова MFA-перевірка або тимчасове блокування облікових записів) — суттєво скорочує вікно можливого компрометації після успішного фішингу.

Канали доставки варіюються: електронна пошта залишається найпоширенішою поверхнею атаки через її універсальність і можливість імперсоналізації. Однак зростає частка смс-фішингу (smishing), голосових атак (vishing), компрометованих акаунтів у месенджерах і соцмережах, фальшивих push-сповіщень або підроблених сторінок постачальників (supply-chain phishing). Drive-by-download та зламані рекламні мережі теж застосовуються як технічні канали доставки, коли фішинг-посилання веде на сайт з експлойтом або на сторінку з формою ловлення облікових даних.

Окрім класичних каналів, атакувальники дедалі частіше використовують комбіновані багатоканальні стратегії — наприклад, спочатку відправляють коротке SMS з посиланням (smishing), а потім надсилають підтверджувальний email або роблять дзвінок (vishing) для підсилення довіри. На мобільних пристроях ризики підвищують фальшиві push-повідомлення й зламані акаунти месенджерів, тому захист мобільного флоту (MDM, контроль встановлення додатків, обмеження deep-linking) має бути невід'ємною частиною стратегії. Компрометація постачальників і підроблені сторінки постачальників (supply-chain phishing) означають, що потрібно моніторити зміни в ланцюжку контрагентів і автоматично перевіряти достовірність повідомлень від партнерів через незалежні канали. Технологічні контрзаходи включають інспекцію URL у пошті (URL rewriting/safe-links), детонацію вкладень у sandbox і браузерну ізоляцію для drive-by сценаріїв, а також репутаційні та поведінкові оцінки доменів у реальному часі. Операційно корисно логувати кліки по посиланнях, прив'язувати їх до сесій користувачів і мати SOAR-процедури, які автоматично блокують домени/URL та ініціюють паролівну ротацію або MFA-перевірку при підозрілих діях. Нарешті, оскільки мобільні і голосові вектори часто супроводжуються техніками соціальної інженерії (як-от SIM-swap або deepfake-дзвінок), важливо поєднувати технічні бар'єри з процесними — наприклад, out-of-band верифікацією транзакцій і політиками повідомлення про підозрілу активність.

Типові «цілі» і payload-и: найпоширеніша мета — викрадення облікових даних (credential harvesting) для подальшого account takeover або lateral movement; інші — встановлення першої точки присутності (initial access) через шкідливі

вкладення або посилання на завантаження, вимагання (ransomware), фінансові шахрайства (BEC — Business Email Compromise) та exfiltration даних. Сучасні кампанії часто поєднують технічні прийоми (малвар-вкладення, одноразові сторінки входу) з людською обробкою (переконати бухгалтера провести платіж). Через це захист повинен закривати і технічний вектор, і процесні слабкі місця.

Індикатори компрометації та ознаки фішингу бувають як очевидними (підозрений домен, невідповідний SPF/DKIM/DMARC результат, незвична IP-адреса відправника), так і тонкими (нетипове час відправлення, невеликі зміни в шаблоні підпису керівника, неспівпадіння Return-Path). Для SOC і адміністрації пошти важливо логувати і корелювати: заголовки Email (Received, From, Return-Path), SPF/DKIM/DMARC статуси, URL-рефери та кліки, поведінку при відкритті вкладень (sandbox execution), та наступні активності в мережі/ендпоінтах, пов'язані з адресатом. Аналіз кліків на URL у sandbox плюс контроль redirect-ланцюгів дають можливість виявляти складніші фішинги.

Технічні заходи захисту: застосування SPF/DKIM/DMARC з політикою відхилення (p=reject) — базова гігієна; поштові шлюзи з Advanced Threat Protection (URL-rewriting, safe-links), ретельний sandboxing вкладень, attachment sandbox detonation, URL-reputation і DLP на вихідні форми. TLS-інспекція і контроль зовнішніх redirect-ів допомагають виявляти C2-поведінку і callback-патерни. Поєднання mail-gateway, EDR і NDR дозволяє корелювати подію фішингу з подальшою поведінкою на ендпоінті й у мережі.

Поведінкові та організаційні контрзаходи — навчання і тестування персоналу, процеси перевірки фінансових транзакцій (dual approval, out-of-band verification), обмеження привілеїв (least privilege) і сегментація фінансових/HR систем. Навчальні кампанії (phishing-simulations) мають бути педагогічними, з фокусом на поясненні помилок, а не на штрафів; важлива регулярність і варіативність сценаріїв, а також відстеження повторних порушників для цільових курсів. Також рекомендовано вводити прості процедури «чи можу я перевірити?» для співробітників, щоб зменшити психологічний тиск при термінових запитах.

Detection engineering для фішингу включає написання правил у SIEM на підозрілу кореляцію: наприклад, якщо на адресу користувача прийшло повідомлення з SPAM-флагом і того ж дня зареєстровано нетипову автозапитувану мережеву активність з невідомого домену, — підняти інцидент. Також корисні спеціалізовані детекції: виявлення незвичних потоків авторизації (аномалії геолокації/часу), підозрілих reset-запитів пароля, багаторазових невдалих входів або одночасних сесій. Автоматизація через SOAR дозволяє: виявивши фішинг-лист — замінити URL у пошті, повідомити користувача, почати масове сканування кліків і, за потреби, ініціювати парольний ресет або мультифакторну перевірку для підозрілих акаунтів.

Інцидент-реагування на фішинг має стандартний workflow: ізоляція та блокування джерела (блок домену/URL, видалення листа з поштових скриньок), негайний збір артефактів (заголовки листа, IP, click-logs, доступні sandbox-репорти), швидка оцінка впливу (чи були розголошені облікові дані? чи виконано файли?), а також форензика ендпоінтів (збір процесних логів, мережових дамів, дамів пам'яті якщо потрібен deeper analysis). Після ліквідації інциденту обов'язковий post-incident review з оновленням детекцій, playbook-ів і, за потреби, повідомленням зовнішніх сторін (регулятори/партнери).

Метрики та контроль ефективності мають включати: click-rate на симульовані фішинги, time-to-report (час від отримання фішингового листа до повідомлення в SOC), частку інцидентів, у яких згадано multi-factor authentication як захист, median Time To Detect (TTD) і median Time To Contain (TTC) для фішингових інцидентів. Ці KPI допомагають пріоритетизувати інвестиції — чи потрібні кращі шлюзи, додаткове навчання чи посилення процесів фінансових перевірок.

Нарешті, важливо слідкувати за еволюцією тактик: з'являються deepfake-аудіо дзвінки у vishing-кампаніях, компрометації поставок (supply-chain phishing), і використання automation/AI для масової персоналізації. Організації мають підтримувати threat-intelligence-стрічки, інтегрувати індикатори з інших секторів (ISAC/CERT) і переглядати політики безпеки у відповідності до нових загроз. Ефективна протидія фішингу — це поєднання технічних бар'єрів, вдосконалених процесів і постійної роботи з людським фактором; тільки такий комплексний підхід системно знижує ризик успішних соціально-інженерних атак.

2.5. ІНСТРУМЕНТИ АНАЛІЗУ ШКІДЛИВОГО ПЗ І ТЕХНОЛОГІЇ ПРОТИДІЇ

Інструменти аналізу шкідливого програмного забезпечення (шкідливого ПЗ) та технології протидії утворюють єдину екосистему: збір багатопланової телеметрії, її глибокий аналіз і автоматизована реакція дозволяють не лише виявляти окремі зразки, а й розуміти методи (Tactics, Techniques and Procedures — тактики, техніки й процедури) супротивника, створюючи репертуар превентивних і коригувальних заходів. У цій взаємопов'язаній моделі ключові блоки — статичний аналіз, динамічний аналіз, аналіз пам'яті (memory forensics), мережева телеметрія, пісочниці (sandbox), інструменти зворотного інжинірингу та системи виявлення і реагування — повинні працювати синхронно, а не окремо, щоб закривати «темні плями» в покритті.

Статичний аналіз — це дослідження бінарного чи скриптового зразка без його запуску. Інструменти статичного аналізу (наприклад, IDA Pro, Ghidra, Binary Ninja, radare2) дозволяють розбирати виконувані файли, вивчати структуру

Portable Executable (PE — формат виконуваного файлу для Windows), таблиці імпорту, цифрові підписи і вбудовані ресурси; проте статичний аналіз слабшає проти сильної обфускації, упаковки й поліморфізму. Тому статичні висновки слід доповнювати метаданими (хеші, ентропія, сертифікати), YARA-патернами (YARA — інструмент для ідентифікації й класифікації зразків за шаблонами) і репортами з інших джерел, щоб сформуванню початкову гіпотезу про походження і швидку категоризацію зразка.

Додатково статичний аналіз часто починається з витягнення рядків (strings) та обчислення хешів і специфічних підписів (наприклад, `imphash` для порівняння таблиць імпорту), а також перевірки ентропії секцій для виявлення упакувальників чи шифрування. Корисно аналізувати метадані файлу — час компіляції, шляхи до PDB, цифрові сертифікати і ланцюги довіри — оскільки ці ознаки часто допомагають зв'язати екземпляри між собою або вказують на використання конкретних збірок/інструментів розробника. Інструменти на кшталт `pefile/lief`, а також утиліти для деобфускації рядків (наприклад, `FLOSS`) і визначення пакерів (`PackerDB/PEiD`-подібні підходи) прискорюють роботу аналітика та дозволяють витягти ранні індикатори. На основі статичного огляду можна сформуванню початковий набір IOC (шляхи файлів, імена mutex, шаблони реєстру, домени/URL у вигляді закодованих рядків) — ці IOC слід негайно прокласифікувати і піддати перевірці у динамічному середовищі. Водночас результати статичного аналізу потрібно корелювати з даними `sandbox`-репортів, EDR- та мережевою телеметрією, оскільки лише така мультидисциплінарна валідація дозволяє перевірити гіпотези про поведінку зразка. Нарешті, всі знахідки треба формалізувати в документовані артефакти для SOC: оновлені YARA-правила, списки IOC, короткі технічні нотатки з припущеннями й обмеженнями — щоб детекції були повторюваними і піддавалися верифікації.

Динамічний аналіз — це запуск зразка в контрольованому середовищі для фіксації його поведінки: створення процесів, звернення до мережі, файлові операції, зміни в реєстрі й виклики API (Application Programming Interface — інтерфейс прикладного програмування). Пісочниці (`sandbox`) автоматизують цю фазу, записуючи послідовності викликів і мережеві взаємодії; їхні звіти дають поведінкові індикатори, корисні для класифікації і написання детекцій. Важливо використовувати різноманітні `sandbox`-платформи і налаштування (емулювання різних операційних систем, різні часові інтервали, рандомізовані середовища), оскільки шкідливі зразки часто містять анти-аналізні механізми, що змінюють поведінку при виявленні дослідницького оточення.

Додатково динамічний аналіз потребує глибокої інструментації: перехоплення API-викликів (`hooking`), трасування системних викликів, моніторингу завантажених модулів, спостереження за створенням/інжекцією потоків і збереженням скріншотів GUI для виявлення тих поведінкових ознак, що

не видно в чистих логах. Важливо одночасно знімати мережевий трафік у форматі pcap, логувати DNS/HTTP-запити та робити TLS-fingerprinting, оскільки багато зразків одразу після запуску встановлюють C2-зв'язок або здійснюють екфільтрацію. Щоб обходити anti-analysis хитрощі (перевірки VM/debugger, таймінгові пастки, очікування користувацької взаємодії), sandbox-и повинні імітувати поведінку користувача (кліки, введення), підтримувати різні таймінги виконання і, де можливо, запускати зразок у повноцінних гостевих ОС, а не емульованих середовищах. Після виконання автоматизований triage порівнює отримані behavioral-features з існуючими базами (sandbox-репорти, threat-intel, TI-fed IOC) і генерує підсумкові артефакти — API-послідовності, network callbacks, dropped-files і memory-dumps — для подальшої класифікації та детекції. Ці артефакти повинні автоматично інтегруватись у SIEM/EDR/NDR, корелюватись і прикриватись до ATT&CK-технік, що дозволяє швидко перетворити поведінкові висновки на Sigma/Kusto-правила і SOAR-playbook-и. І нарешті, слід пам'ятати про обмеження: навіть найкращі sandbox-и іноді пропускають таргетовані або time-bomb зразки, тому динамічний аналіз має поєднуватись зі статикою, memory-forensics і ручною перевіркою аналітиком для надійної валідації гіпотез.

Аналіз оперативної пам'яті (memory forensics) дає найцінніші артефакти у випадках обфускації та безфайлового виконання: ключі шифрування, розпакований код, інститути внутрішніх структур і тимчасові індикатори. Інструменти на кшталт Volatility або Rekall призначені для витягання артефактів із дамів пам'яті і відновлення ланцюжків виконань; ці дані особливо критичні, коли на диску немає явних слідів. Тому оперативна політика реагування має включати автоматичні тригери на збирання memory-dump при підозрі, з дотриманням правил збереження ланцюжка зберігання доказів (chain-of-custody) для юридично значущих розслідувань.

Додатково важливо розрізняти типи знімків пам'яті — повний дам оперативної пам'яті хоста, дам окремого процесу або знімки гіпервізора/VM — оскільки кожен дає різну глибину артефактів (kernel-артефакти та інжектвані модулі зазвичай видно тільки у повному дампі).

У процесі аналізу з дамів пам'яті виконують «carving» артефактів (витяг рядків, бінарних фрагментів, TLS-контекстів), пошук ключів шифрування і конфігураційних структур, а також застосовують YARA-сканування по пам'яті для виявлення відомих шаблонів, які не видно на диску. Memory-forensics дозволяє виявляти інжекції коду (инъекции DLL, reflective loading), патерни розпаковування в оперативній пам'яті та тимчасові мережеві контексти (відкриті сокети, незавершені TLS-сесії), що допомагає відтворити ланцюжок виконань і часову лінію компрометації.

Кореляція витягнутих memory-артефактів із EDR-лозами (ProcessCreate, ImageLoad, CreateRemoteThread) і мережевими дампами (pcap, DNS) підвищує

достовірність висновків і дозволяє відокремити справжні індикатори компрометації від шуму.

З оперативної точки зору необхідно автоматизувати тригери збору пам'яті через SOAR-плейбуки (збирання повного дампу, експорт відповідних журналів і завантаження артефактів у аналізатор), але з урахуванням впливу на продуктивність і з дотриманням юридичних процедур щодо збереження доказів.

Слід також враховувати обмеження та складнощі: великі обсяги даних потребують підготовленої інфраструктури зберігання і обробки, частина пам'яті може бути зашифрована або замаскована anti-forensic прийомами, тому план реагування має передбачати пріоритезацію хостів, використання snapshot-методів віртуалізації та тісну співпрацю з юридичним відділом для коректної обробки зібраних доказів.

Мережева телеметрія — логуювання DNS-запитів, HTTP(S)-трафіку, потоків NetFlow/IPFIX, TLS-отпечатків (TLS-fingerprinting) і проксі-логів — дозволяє виявляти командно-контрольні (C2 — Command and Control) канали, екфільтрацію та нетипові callback-поведінки. Системи Network Detection and Response (NDR — виявлення і реагування в мережі) такі як Zeek (раніше Bro), Suricata або комерційні платформи корисні для підвищення видимості; кореляція мережових і ендпоінтних сигналів в SIEM (Security Information and Event Management — система керування інформацією і подіями безпеки) дає контекст і зменшує кількість хибних спрацьовувань.

Крім перерахованих джерел, корисно фокусуватися на виявленні специфічних мережових патернів — DNS-тунелювання (нестандартні доменні запити, аномальна довжина запиту, часті піддомени), регулярні beacon-інтервали (таймінги callback-ів) та аномалії у User-Agent/Referrer строках для HTTP. TLS-fingerprinting (наприклад, JA3/JA3S) дає можливість відокремити легітимні зашифровані з'єднання від підозрілих клієнтських стеків, а аналіз SNI/ALPN у TLS-handshake допомагає виявляти фальшиві або схожі домени навіть при використанні HTTPS. Збір pcap і NetFlow/ IPFIX корисний для ретроспективного розслідування — дозволяє реконструювати маршрути екфільтрації, виявити пікові періоди передачі даних і зіставити їх з подіями на ендпоінтах.

Сучасні NDR-платформи використовують поведінкові моделі й ML-детектори для виявлення аномалій на рівні потоку, але їхня ефективність значно зростає при кореляції з EDR-сигналами (наприклад, розпаковування в пам'яті + одразу після цього outbound-з'єднання). Автоматизація реакцій через SIEM/SOAR (блокування IP/домени, ізоляція сегмента мережі, оповіщення відповідальних) скорочує час на containment і дає змогу швидко розгорнути контрзаходи.

Необхідно також пам'ятати про обмеження: шифрування трафіку, використання політик split-tunneling у VPN і зростання трафіку в хмарних середовищах ускладнюють видимість, тому варто інвестувати в TLS-інспекцію там,

де це допустимо, та у внутрішню телеметрію (mirror/span, cloud flow logs, host-based telemetry) для покриття «темних плям».

Інструменти зворотного інжинірингу (reverse engineering) і статично-динамічний гібридний аналіз залишаються серцем технічного розслідування: дизасемблери, дизасемблери з графічними інтерфейсами, дизасемблери для скриптів, а також інтегровані середовища для аналізу мережевої поведінки і API-викликів дозволяють аналітику створити докладну технічну карту (ТТР — Tactics, Techniques and Procedures). Водночас на рівні операцій важливо не застрягати в глибокій техніці для одного зразка: результати інженерної роботи мають трансформуватися в repeatable (повторювані) правила та playbook-и для SOC (Security Operations Center — центр операцій безпеки).

Для цього аналітики зворотного інжинірингу повинні тісно співпрацювати з detection-engineering: технічні висновки (ідентифікатори коду, характерні API-послідовності, конфігураційні шаблони) повинні перетворюватися на конкретні артефакти — YARA-правила, Sigma-детекції, Kusto/Elastic-запити та NDR-сигнатури. Використання сучасних інструментів динамічної інструментації (наприклад, Frida, DynamoRIO, PIN) і трасування API дозволяє витягти повторювані поведінкові патерни, які легше протестувати в автоматизованому середовищі, ніж байтові сигнатури. Впровадження CI/CD-процесів для детекцій: кожне нове правило має проходити регресійні тести проти відомих benign/malicious наборів, метрик false-positive і production-даних перед деплоєм у SOC. Створення бібліотеки «сімейних» підписів і тестових кейсів (sandbox-репорти, memory-dumps, pcap) прискорює перевірку гіпотез і дозволяє відтворювати результати між аналітиками. Не менш важливо документувати припущення, межі застосування та очікувані джерела телеметрії для кожної детекції, щоб аналітики SOC могли швидко верифікувати спрацювання і зрозуміти причину false-positive. І нарешті, налагоджений зворотний зв'язок між SOC, threat-intel і реверс-командою — регулярні post-mortem, оновлення ATT&CK-мапінгів та метрики TTD/TTC — забезпечують безперервне підвищення якості правил і зниження ризику пропуску нових варіантів шкідливого ПЗ.

Detection engineering (інженерія виявлення) і threat hunting (полювання на загрози) — практики, які перетворюють технічні висновки на робочі детекції. Тут використовують YARA-правила для файлових і пам'ятних шаблонів, Sigma (універсальний формат для опису детекцій у SIEM), Kusto Query Language (для Microsoft Sentinel) або Elasticsearch Query DSL, а також машинне навчання на основі поведінкових фіч (частота API-викликів, таймінги beaconing-у, аномалії командних рядків). Ключове правило — кожна нова детекція має документуватися

з припущеннями, джерелами даних і перевітками на історичних логах, щоб мінімізувати false positives (помилкові спрацьовування).

Детекшн-інженерія вимагає чіткого мапінгу кожного правила на конкретні джерела телеметрії — Endpoint Detection and Response (EDR), Network Detection and Response (NDR), проксі-сервери, cloud-логи — і на відповідні техніки MITRE ATT&CK, щоб розуміти «темні плями» в покритті. Необхідно впровадити тестування та CI/CD-пайплайни для детекцій: автоматизовані регресійні тести проти маркованих наборів benign і malicious, інжекти синтетичного трафіку/подій і вимірювання шуму (noise) перед розгортанням у продакшн.

Тонінг порогів та enrichment (збагачення) подій має базуватись на threat-intelligence (розвідці загроз) — підключенні фідів з індикаторами компрометації (Indicators of Compromise, IOC) і даних про Tactics, Techniques and Procedures (TTP) разом із контекстом критичності активу й поведінковими базелайнами користувачів, щоб зменшити кількість false positives (помилкових спрацьовувань). Threat hunting (полювання на загрози) повинен бути гіпотезно-орієнтованим і документованим: кожна гіпотеза — джерела даних, запити, очікувані індикатори й результати — має конвертуватися в repeatable playbook-и та у SOAR (Security Orchestration, Automation and Response) runbook-и для автоматичного опрацювання.

Потрібно встановити чіткі KPI і замкнуті цикли зворотного зв'язку — median Time To Detect (медіанний час виявлення), median Time To Contain (медіанний час утримання), частка false positives і відсоток coverage (покриття технік) — і вимагати пост-деплой моніторингу та коригувань правил за результатами.

Регулярні purple-team і red-team вправи повинні перевіряти реальність детекцій у бойових умовах та пріоритизувати інженерні завдання, а тісна співпраця threat-intel, detection-engineering і Security Operations Center (SOC) гарантує, що виявлення залишаються ефективними проти еволюціонуючих загроз.

Технології протидії включають превентивні (patch-менеджмент, Application Control / Whitelisting — наприклад, Microsoft AppLocker або Windows Defender Application Control), детекційні (Endpoint Detection and Response — EDR, SIEM, NDR) і реактивні (Security Orchestration, Automation and Response — SOAR — оркестрація, автоматизація та реагування). Автоматизація через SOAR дозволяє визначеним playbook-ам виконувати рутинні дії (ізоляція хоста, збір forensic-артефактів, блокування доменів), скорочуючи час від виявлення до containment (утримання) і recovery (відновлення). Водночас технології — це лише частина рішення: процеси, політики і підготовка людей (training) вирішують, чи зможе організація ефективно використати ці інструменти.

Ефективна архітектура захисту будується на інтеграції: EDR дає процесну і пам'ятну видимість, NDR — мережевий контекст, SIEM — кореляцію і збереження історії, а threat-intelligence (розвідка загроз) постачає індикатори компрометації (Indicators of Compromise — IOC) і TTP, які постійно оновлюють правила. Практика

map-інгу на MITRE ATT&CK (Adversarial Tactics, Techniques, and Common Knowledge — тактики, техніки та процедурні відомості) допомагає стандартизувати опис атак і побудувати coverage-matrix (матрицю покриття), щоб відстежувати «темні плями» у телеметрії і пріоритетизувати інвестиції.

Крім інструментальної інтеграції, ефективна стратегія захисту вимагає чіткої оркестрації політик — наприклад, централізованого управління конфігураціями, моніторингу відхилень (config drift) і автоматичних перевірок безпечних налаштувань для хоста, мережі та хмарних сервісів. Не менш важливими є заходи з контролю доступу: сильна ідентифікація та аутентифікація (IAM), обов'язкове застосування багатофакторної аутентифікації (MFA), принцип найменших привілеїв та регулярний рев'ю прав доступу з логами аудиту. У хмарних і контейнеризованих середовищах слід інтегрувати telemetry на рівні платформи (cloud flow logs, audit trails, runtime security) і застосовувати IaC-сканування та контейнерні політики безпеки для виявлення вразливих конфігурацій ще на етапі CI/CD. Управління ризиком постачальників (third-party risk) і перевірки ланцюга поставок мають стати частиною процесу закупівель та контрактів, щоб мінімізувати ризики від сторонніх компонентів. Для обґрунтування інвестицій і прийняття рішень корисно зв'язувати технічні метрики з бізнес-результатами — наприклад, прогнозоване зниження median TTD/TTC, зменшення кількості інцидентів проти критичних сервісів і економія на витратах на інцидент-реагування. Нарешті, регулярні вправи (tabletop, red/purple-team, incident simulations), навчання персоналу і постійний цикл покращень через post-mortem гарантують, що технології, процеси й люди працюють узгоджено й організація не лише має інструменти, а й уміє ними ефективно користуватися.

Окремий блок — правові й етичні аспекти аналізу: збирання дамів пам'яті, копіювання користувацьких даних та виконання емуляції в середовищі може зачіпати питання приватності і регуляторні вимоги; тому SOC і IR (Incident Response — реагування на інциденти) команди мають узгоджувати дії з юридичним відділом і виконувати процедури збереження ланцюга зберігання доказів (chain-of-custody). Також важливо мати стандартизовані SLAs (Service Level Agreements — угоди про рівень сервісу) на реагування і узгоджені KPI (Key Performance Indicators — ключові показники ефективності), наприклад, coverage % (відсоток покриття технік), median Time To Detect (TTD — медіанний час виявлення) і median Time To Contain (TTC — медіанний час утримання).

Нарешті, для практичної готовності необхідна постійна верифікація: регулярні red-team / purple-team вправи (імітація атак і спільна робота атакуючих і захисників), тестування playbook-ів, оновлення YARA/Sigma правил і періодичне пере-навчання ML-моделей. Лише за умови, що інструменти аналізу, технології протидії, процеси і люди працюють узгоджено, організація отримає реальну

можливість виявляти, реагувати й знижувати ризики від шкідливого програмного забезпечення.

СПИСОК ДЖЕРЕЛ РОЗДІЛУ 2

1. Костюк Ю.В., Складанний П.М., Бебешко Б.Т., Хорольська К.В., Рзаєва С.Л., Ворохоб М.В. (2025) Безпека інформаційно-комунікаційних систем. Київський університет імені Бориса Грінченка, Україна, Київ. - 1017 с.
2. Костюк Ю.В., Складанний П.М., Гулак Г.М., Бебешко Б.Т., Хорольська К.В., Рзаєва С.Л. (2025) Системи захисту інформації. Київський університет імені Бориса Грінченка, Україна, Київ. - 886с.
3. Kostiuk, Yu. V., Skladannyi, P. M., Bebeshko, B. T., Khorolska, K. V., Rzaieva, S. L., & Vorokhob, M. V. (2025). Information and communication systems security. [Textbook] Kyiv: Borys Grinchenko Kyiv Metropolitan University.
4. Kostiuk, Yu. V., Skladannyi, P. M., Hulak, H. M., Bebeshko, B. T., Khorolska, K. V., & Rzaieva, S. L. (2025). Information security systems. [Textbook] Kyiv: Borys Grinchenko Kyiv Metropolitan University.
5. Гулак Г. М., Жильцов О. Б., Киричок Р. В., Коршун Н. В., Складанний П. М. Інформаційна та кібернетична безпека підприємства : підруч. / Г. М. Гулак, О. Б. Жильцов, Р. В. Киричок, Н. В. Коршун, П. М. Складанний – Львів : Видавець Марченко Т. В., 2023. – 370 с.
6. Крючкова, Л., Яремчук, Д., & Пазинін, А. (2024). Планування навантаження контейнерів за допомогою лінійного програмування. Телекомунікаційні та інформаційні технології, 2(83), 114 –119. <https://doi.org/10.31673/2412-4338.2024.022935>
7. Цирканюк, Д., & Соколов, В. (2024). Методика розслідування інцидентів інформаційної безпеки. Кібербезпека: освіта, наука, техніка, 2(26), 140–154. <https://doi.org/10.28925/2663-4023.2024.26.675>
8. Козачок, В., & Драпатий, М. (2024). Аналіз технології розслідування інцидентів безпеки на об'єктах критичної інфраструктури. Кібербезпека: освіта, наука, техніка, 2(26), 374–391. <https://doi.org/10.28925/2663-4023.2024.26.699>
9. Шевченко, С., Жданова, Ю., Складанний, П., & Петренко, Т. (2024). Нечіткі когнітивні карти як інструмент візуалізації сценаріїв реагування на інциденти в системах безпеки. Кібербезпека: освіта, наука, техніка, 2(26), 417–429. <https://doi.org/10.28925/2663-4023.2024.26.707>
10. Корнієць, В., & Жданова, Ю. (2024). Методика формування швидкісного імітостійкого шифру багато алфавітної заміни. Кібербезпека:

освіта, наука, техніка, 2(26), 476–486. <https://doi.org/10.28925/2663-4023.2024.26.723>

11. Руда, Х., Сабодашко, Д., Микитин, Г., Швед, М., Бордуляк, С., & Коршун, Н. (2024). Порівняння методів цифрової обробки сигналів та моделей глибинного навчання у голосовій аутентифікації. *Кібербезпека: освіта, наука, техніка*, 1(25), 140–160. <https://doi.org/10.28925/2663-4023.2024.25.140160>

12. Жданова, Ю., Шевченко, С., Спасітелєва, С., & Сокульський, О. (2024). Прийняття рішень на основі лінійної оптимізації у процесі управління ризиками інформаційної безпеки. *Кібербезпека: освіта, наука, техніка*, 1(25), 330–343. <https://doi.org/10.28925/2663-4023.2024.25.330343>

13. Довженко, Н., Іваніченко, Є., Складанний, П., & Аушева, Н. (2024). Інтеграція безпеки та відмовостійкості сенсорних мереж на основі аналізу енергоспоживання та трафіку. *Кібербезпека: освіта, наука, техніка*, 1(25), 390–400. <https://doi.org/10.28925/2663-4023.2024.25.390400>

14. Іосіфов, Є., & Соколов, В. (2024). Порівняльний аналіз методів, технологій, сервісів та платформ для розпізнавання голосової інформації в системах забезпечення інформаційної безпеки. *Кібербезпека: освіта, наука, техніка*, 1(25), 468–486. <https://doi.org/10.28925/2663-4023.2024.25.468486>

15. Барабаш, О., Аушева, Н., Складанний, П., Іваніченко, Є., & Довженко, Н. (2024). Технічні аспекти побудови відмовостійкої інфраструктури сенсорної мережі. *Кібербезпека: освіта, наука, техніка*, 4(24), 185–195. <https://doi.org/10.28925/2663-4023.2024.24.185195>

16. Крючкова, Л., & Леонтюк, Н. (2024). Програмно-апаратна реалізація алгоритму швидкої оцінки потужності Wi-Fi сигналу в точках простору урбанізованого приміщення. *Кібербезпека: освіта, наука, техніка*, 4(24), 241–256. <https://doi.org/10.28925/2663-4023.2024.24.241256>

17. Крючкова, Л., & Дяків, В. (2024). Програмна реалізація розрахунку просторового розподілу поля рамок антен. *Кібербезпека: освіта, наука, техніка*, 4(24), 312–320. <https://doi.org/10.28925/2663-4023.2024.24.312320>

18. Іосіфов, Є., & Соколов, В. (2024). Методи аналізу природної мови та застосування нейронних мереж в кібербезпеці. *Кібербезпека: освіта, наука, техніка*, 4(24), 398–414. <https://doi.org/10.28925/2663-4023.2024.24.398414>

19. Шевченко, С., Жданова, Ю., Спасітелєва, С., Мазур, Н., Складанний, П., & Негоденко, В. (2024). Математичні методи в кібербезпеці: кластерний аналіз та його застосування в інформаційній та кібернетичній безпеці. *Кібербезпека: освіта, наука, техніка*, 3(23), 258–273. <https://doi.org/10.28925/2663-4023.2024.23.258273>

20. Крючкова, Л., & Ємельяненко, М. (2024). Захист web-ресурсу intranet від зовнішніх і внутрішніх загроз. *Кібербезпека: освіта, наука, техніка*, 3(23), 318–327. <https://doi.org/10.28925/2663-4023.2024.23.318327>

21. Крючкова, Л., & Стеблина, О. (2024). Захист смартфонів від впливу шкідливих програм в процесі зарядки у громадських місцях. *Кібербезпека: освіта, наука, техніка*, 3(23), 338–347. <https://doi.org/10.28925/2663-4023.2024.23.338347>
22. Скіцько, О., Складанний, П., Ширшов, Р., Гуменюк, М., & Ворохоб, М. (2023). Загрози та ризики використання штучного інтелекту. *Кібербезпека: освіта, наука, техніка*, 2(22), 6–18. <https://doi.org/10.28925/2663-4023.2023.22.618>
23. Кіпчук, Ф., & Соколов, В. (2023). Модель розрахунку витрат на баг-баунті програми тестування вразливостей безпеки. *Кібербезпека: освіта, наука, техніка*, 2(22), 68–83. <https://doi.org/10.28925/2663-4023.2023.22.6883>
24. Марценюк, М., Козачок, В., Богданов, О., Іосіфов, Є., & Бржевська, З. (2023). Аналіз методів виявлення дезінформації в соціальних мережах за допомогою машинного навчання. *Кібербезпека: освіта, наука, техніка*, 2(22), 148–155. <https://doi.org/10.28925/2663-4023.2023.22.148155>
25. Машталяр, Я., Козачок, В., Бржевська, З., Богданов, О., Оксанич, І., & Литвинов, В. (2023). Дослідження розвитку та інновації кіберзахисту на об'єктах критичної інфраструктури. *Кібербезпека: освіта, наука, техніка*, 2(22), 156–167. <https://doi.org/10.28925/2663-4023.2023.22.156167>
26. Шевченко, С., Жданова, Ю., Складанний, П., & Бойко, С. (2023). Теоретико-ігровий підхід до моделювання конфліктів у системах інформаційної безпеки. *Кібербезпека: освіта, наука, техніка*, 2(22), 168–178. <https://doi.org/10.28925/2663-4023.2023.22.168178>
27. Соколов, В., & Складанний, П. (2023). Методика оцінки комплексних збитків від інциденту інформаційної безпеки. *Кібербезпека: освіта, наука, техніка*, 1(21), 99–120. <https://doi.org/10.28925/2663-4023.2023.21.99120>
28. Спасітелєва, С., Чичкань, І., Шевченко, С., & Жданова, Ю. (2023). Розробка безпечних контейнерних застосунків з мікросервісною архітектурою. *Кібербезпека: освіта, наука, техніка*, 1(21), 193–210. <https://doi.org/10.28925/2663-4023.2023.21.193210>
29. Костюк Ю., Бебешко Б., Складанний П., Рзаєва С., Хорольська К. Оптимізація буфера та пріоритетів для забезпечення безпеки У Bluetooth-мережах. *Безпека інформаційних систем і технологій*. 2025. 2(8), 5-16.
30. Ворохоб, М., Киричок, Р., Яскевич, В., Добришин, Ю., & Сидоренко, С. (2023). Сучасні перспективи застосування концепції zero trust при побудові політики інформаційної безпеки підприємства. *Кібербезпека: освіта, наука, техніка*, 1(21), 223–233. <https://doi.org/10.28925/2663-4023.2023.21.223233>

31. Добришин, Ю., Сидоренко, С., & Ворохоб, М. (2023). Автоматизована система підтримки прийняття рішення щодо відновлення пошкодженого програмного забезпечення внаслідок впливу кібератак. *Кібербезпека: освіта, наука, техніка*, 4(20), 174–182. [https://doi.org/10.28925/2663-4023.20.174182](https://doi.org/10.28925/2663-4023.2023.20.174182)
32. Соколов, В., & Складанний, П. (2023). Порівняльний аналіз стратегій побудови другого та третього рівня освітніх програм зі спеціальності 125 «Кібербезпека». *Кібербезпека: освіта, наука, техніка*, 4(20), 183–204. [https://doi.org/10.28925/2663-4023.20.183204](https://doi.org/10.28925/2663-4023.2023.20.183204)
33. Shevchenko, S., Zhdanova, Y., & Spasiteleva, S. (2023). Mathematical methods in cybersecurity: Catastrophe theory. *Cybersecurity: Education, Science, Technique*, 3(19), 165–175. <https://doi.org/10.28925/2663-4023.2023.19.165175>
34. Sokolov, V. (2022). Approaches to the formation of scientific thinking in cybersecurity high school students. *Cybersecurity: Education, Science, Technique*, 2(18), 124–137. <https://doi.org/10.28925/2663-4023.2022.18.124137>
35. Shevchenko, S., Skladannyi, P., Nehodenko, O., & Nehodenko, V. (2022). Study of applied aspects of conflict theory in security systems. *Cybersecurity: Education, Science, Technique*, 2(18), 150–162. <https://doi.org/10.28925/2663-4023.2022.18.150162>
36. Sukaylo, I., & Korshun, N. (2022). The influence of NLU and generative AI on the development of cyber defense systems. *Cybersecurity: Education, Science, Technique*, 2(18), 187–196. <https://doi.org/10.28925/2663-4023.2022.18.187196>
37. Костюк Ю., Бебешко Б., Гулак Г., Складанний П., Рзаєва С., Хорольська К. (2024) Забезпечення кібербезпеки та швидкодії передачі даних у безпроводних мережах. *Безпека інформації*, т. 30, №3, 365-375.
38. Hulak, H., Zhdanova, Y., Skladannyi, P., Hulak, Y., & Korniiets, V. (2022). Vulnerabilities of short message encryption in mobile information and communication systems of critical infrastructure objects. *Cybersecurity: Education, Science, Technique*, 1(17), 145–158. <https://doi.org/10.28925/2663-4023.2022.17.145158>
39. Romaniuk, O., Skladannyi, P., & Shevchenko, S. (2022). Comparative analysis of solutions to provide control and management of privileged access in the IT environment. *Cybersecurity: Education, Science, Technique*, 4(16), 98–112. <https://doi.org/10.28925/2663-4023.2022.16.98112>
40. Borkivska, O., Zhurakovskiy, B., & Platonenko, A. (2022). Smart city information system based on LoRa technology. *Cybersecurity: Education, Science, Technique*, 4(16), 113–128. <https://doi.org/10.28925/2663-4023.2022.16.113128>
41. Kostiuk, Y. V., Skladannyi, P. M., Hulak, H. M., Bebeshko, B. T., Khorolska, K. V., & Rzaieva, S. L. (2025). *Information security systems.*[Textbook] Kyiv: Borys Grinchenko Kyiv Metropolitan University.

42. Brzhevskaya, Z., Kyrychok, R., Platonenko, A., & Hulak, H. (2022). Assessment of the preconditions of formation of the methodology of assessment of information reliability. *Cybersecurity: Education, Science, Technique*, 3(15), 164–174. <https://doi.org/10.28925/2663-4023.2022.15.164174>
43. Shevchenko, S., Zhdanova, Y., Skladannyi, P., & Boiko, S. (2022). Insiders and insider information: Essence, threats, activities and legal responsibility. *Cybersecurity: Education, Science, Technique*, 3(15), 175–185. <https://doi.org/10.28925/2663-4023.2022.15.175185>
44. Довженко, Н., Мазур, Н., Костюк, Ю., Рзаєва, С. Інтеграція ІОТ ТА штучного інтелекту в інтелектуальні транспортні системи. *Кібербезпека: освіта, наука, техніка*. 2024. 2(26), 430–444.
45. Bogachuk, V. Sokolov and V. Buriachok, "Monitoring Subsystem for Wireless Systems Based on Miniature Spectrum Analyzers," *2018 International Scientific-Practical Conference Problems of Infocommunications. Science and Technology (PIC S&T)*, Kharkiv, Ukraine, 2018, pp. 581-585, doi: 10.1109/INFOCOMMST.2018.8632151.
46. Kostiuk, Y. V., Skladannyi, P. M., Bebesko, B. T., Khorolska, K. V., Rzaieva, S. L., & Vorokhob, M. V. (2025). Information and communication systems security.[Textbook] Kyiv: Borys Grinchenko Kyiv Metropolitan University.
47. Hulak, Hennadii and Skladannyi, Pavlo and Sokolov, Volodymyr and Hulak, E. and Korniiets, V. (2022) *Dynamic model of guarantee capacity and cyber security management in the critical automated systems* 2nd International Conference on Conflict Management in Global Information Networks (CMiGiN 2022), 3530. pp. 102-111. ISSN 1613-0073
48. Sokolov, V. Y. and Skladannyi, Pavlo and Hulak, Hennadii (2022) *Stability Verification of Self-Organized Wireless Networks with Block Encryption* Cybersecurity Providing in Information and Telecommunication Systems, 3137. pp. 227-237. ISSN 1613-0073
49. Anakhov, P. and Zhebka, V. and Popereshnyak, S and Skladannyi, Pavlo and Sokolov, Volodymyr (2023) *Protecting Objects of Critical Information Infrastructure from Wartime Cyber Attacks by Decentralizing the Telecommunications Network* Cybersecurity Providing in Information and Telecommunication Systems, 3550. pp. 240-245. ISSN 1613-0073
50. Костюк Ю., Довженко, Н., Мазур, Н., Складанний, П., Рзаєва, С. Методика захисту GRID-середовища від шкідливого коду під час виконання обчислювальних завдань. *Кібербезпека: освіта, наука, техніка*. 2025. 3(27), 22–40.
51. Kipchuk, F. and Sokolov, Volodymyr and Skladannyi, Pavlo and Zhylytsov, Oleksii and Ageyev, D. (2023) *Method for Increasing the Various Sources Data Consistency for IoT Sensors* IEEE 9th International

Conference on Problems of Infocommunications, Science and Technology (PICST). pp. 522-526. ISSN 979-8-3503-9891-5

52. A. Carlsson, D. Ageyev, Y. Sadykov and V. Sokolov, "Sustainability Research of the Secure Wireless Communication System with Channel Reservation," *2020 IEEE 15th International Conference on Advanced Trends in Radioelectronics, Telecommunications and Computer Engineering (TCSET)*, Lviv-Slavske, Ukraine, 2020, pp. 973-977, doi: 10.1109/TCSET49122.2020.235583.

53. Складанний, П., Костюк Ю., Рзаєва, С., Самойленко, Ю., Савченко, Т. Розробка модульних нейронних мереж для виявлення різних класів мережевих атак. *Кібербезпека: освіта, наука, техніка*. 2025. 3(27), 534–548.

54. F. Kipchuk, V. Sokolov, V. Buriachok and L. Kuzmenko, "Investigation of Availability of Wireless Access Points based on Embedded Systems," *2019 IEEE International Scientific-Practical Conference Problems of Infocommunications, Science and Technology (PIC S&T)*, Kyiv, Ukraine, 2019, pp. 1-5, doi: 10.1109/PICST47496.2019.9061551.

55. Shevchuk, D and Harasymchuk, O and Partyka, A and Korshun, Natalia (2023) *Designing Secured Services for Authentication, Authorization, and Accounting of Users* Cybersecurity Providing in Information and Telecommunication Systems, 3550. pp. 207-225. ISSN 1613-0073

56. Складанний П.М., Костюк Ю.В., Мазур Н.П., Пітайчук М.А. Дослідження характеристик та продуктивності протоколів доступу до хмарних обчислювальних середовищ на основі універсального тестування. *Телекомунікаційні та інформаційні технології*. 2025. №1(86). С. 61-74.

КОНТРОЛЬНІ ПИТАННЯ

1. Які основні класи шкідливого програмного забезпечення (malware) згадані в розділі, і чим відрізняються віруси від троянських програм (trojans)?

2. Поясніть модель «Ланцюг кібернападу (Cyber Kill Chain)» за Lockheed Martin і як її поєднувати з MITRE ATT&CK (Adversarial Tactics, Techniques, and Common Knowledge — тактики, техніки та процедурні відомості) для побудови timeline інциденту.

3. Які конкретні техніки ухилення та приховування атак описані (обфускація, поліморфізм, метаморфізм, fileless) і чому вони ускладнюють статичний аналіз?

4. Які етапи розвідки (reconnaissance) зазвичай використовують зловмисники при підготовці фішингової кампанії, і які OSINT-джерела для цього згадані?

5. Назвіть і поясніть основні канали доставки фішингових та соціально-інженерних атак (email, smishing, vishing, drive-by, supply-chain phishing).

6. Які оперативні та технічні контрзаходи рекомендовано застосовувати проти фішингу (SPF/DKIM/DMARC, sandboxing, URL-rewriting, SOAR-автоматизація тощо)? Розпишіть коротко.

7. Що таке статичний аналіз шкідливого ПЗ, які інструменти для нього згадані та які його ключові обмеження?

8. У чому суть динамічного аналізу (sandboxing), які типи артефактів він генерує і яких anti-analysis прийомів слід очікувати від зразків?

9. Чому memory forensics (аналіз оперативної пам'яті) важливий при fileless-атаках і які інструменти для цього рекомендуються (наприклад, Volatility / Rekall)?

10. Які мережеві джерела телеметрії (DNS, HTTP(S), NetFlow/IPFIX, TLS-fingerprinting) корисні для виявлення C2-каналів і ексфільтрації, і як їх корелювати з ендпоінтною телеметрією?

11. Що таке EDR (Endpoint Detection and Response — виявлення і реагування на кінцевих точках), NDR (Network Detection and Response — виявлення і реагування в мережі), SIEM (Security Information and Event Management — управління інформацією та подіями безпеки) і SOAR (Security Orchestration, Automation and Response — оркестрація, автоматизація та реагування на інциденти), і яку роль вони грають у архітектурі захисту?

12. Опишіть, як результати зворотного інжинірингу (reverse engineering) слід трансформувати в repeatable (повторювані) детекції (YARA, Sigma, Kusto/Elastic-запити) і SOAR-playbook-и.

13. Які ключові етапи IR-плану (Incident Response — реагування на інциденти) для випадку шкідливого ПЗ і які дії входять до першої години реагування (short checklist)?

14. Які особливості виявлення і реагування на руткіти (rootkits), зокрема kernel-mode та bootkit/UEFI-варіанти, і які апаратні механізми захисту рекомендуються (Secure Boot, TPM, Measured Boot)?

15. Які архітектури ботнетів існують (централізовані, P2P, гібридні), які механізми стійкості вони використовують (DGA, fast-flux), і які мережеві заходи нейтралізації ефективні (sinkholing, takedown)?

16. У чому полягає модель RaaS (Ransomware-as-a-Service) і які конкретні заходи зменшують ризик успішної операції програм-вимагачів (ransomware) проти організації?

17. Які метрики ефективності реагування на інциденти згадані в тексті (Time To Detect — TTD, Time To Contain — TTC, Time To Remediate/Recover — TTR) і як вони використовуються для пріоритизації інвестицій у захист?

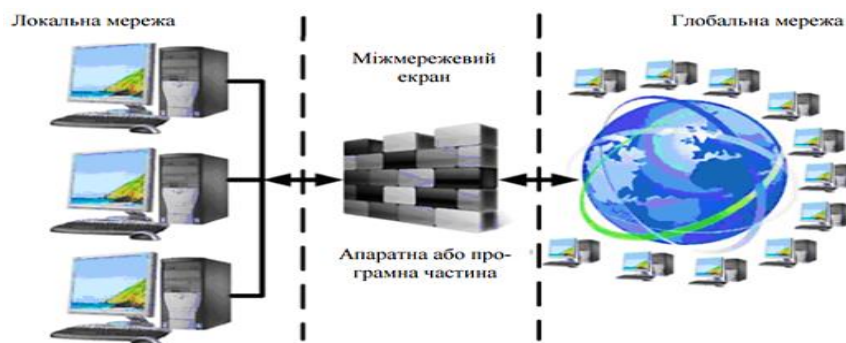
18. Опишіть процес «coverage-mapping» щодо MITRE ATT&CK: які дані потрібні, як визначити «темні плями» у телеметрії і як пріоритизувати інженерні зусилля.

19. Які юридичні та організаційні аспекти необхідно враховувати під час форензики і повідомлення про інцидент (chain-of-custody, співпраця з Legal, повідомлення регуляторів і PR-координація)?

20. Наведіть приклад практичної вправи (tabletop / purple-team / red-team) для перевірки готовності: яка мета вправи, які сценарії слід відпрацювати і які KPI вимірювати після завершення?

РОЗДІЛ 3

ЗАХИСТ КОМП'ЮТЕРНИХ МЕРЕЖ ВІД АТАК: МОДЕЛІ ЗАГРОЗ, ВРАЗЛИВОСТІ НА РІЗНИХ РІВНЯХ СТЕКУ ПРОТОКОЛІВ ТА СУЧАСНІ МЕРЕЖЕВІ ТЕХНОЛОГІЇ БЕЗПЕКИ



- 3.1. Моделі атакувальників і загроз: від індивідуальних хакерів до державних АРТ-груп**
- 3.2. Атаки на мережевому рівні: DDoS, ARP-Spoofing, IP-Spoofing**
- 3.3. Атаки на транспортному рівні: TCP Hijacking, UDP Flooding, SYN Flood**
- 3.4. Атаки на прикладному рівні: SQL Injection, XSS, CSRF, Command Injection**
- 3.5. Протоколи та механізми захисту мереж: VPN, IPsec, MPLS, Zero Trust Network Access**
- 3.6. Виклики безпеки у хмарних та SDN-мережах**
- 3.7. Вразливості та експлойти: від помилок конфігурації до Zero-Day атак**
- 3.8. Інструменти моніторингу й виявлення атак: IDS/IPS, SIEM, EDR, NDR**

Захист мереж ґрунтується на моделях загроз (threat models), які визначають атакуючих, їхні мотиви та ресурси, та на ризик-орієнтованій пріоритизації активів. Вразливості слід розглядати по рівнях стеку протоколів (модель OSI — Open Systems Interconnection): від фізичного

рівня до прикладного, де проявляються атаки типу ARP-spoofing (ARP — Address Resolution Protocol), BGP-hijacking (BGP — Border Gateway Protocol), DDoS (Distributed Denial of Service — розподілений відмова в обслуговуванні), ін'єкції SQL/XSS і вразливості API (Application Programming Interface).

Сучасна захисна архітектура — багаторівнева і оркестрована: превентивні заходи (patch-менеджмент, application control / whitelisting, Multi-Factor Authentication — MFA), детекція (EDR — Endpoint Detection and Response, NDR — Network Detection and Response, SIEM — Security Information and Event Management) і реакція з автоматизацією (SOAR — Security Orchestration, Automation and Response). Технології Zero Trust Network Access (ZTNA) і Secure Access Service Edge (SASE) поєднують контроль доступу, SD-WAN (Software-Defined Wide Area Network) і хмарні сервіси безпеки; у хмарі використовуються CSPM (Cloud Security Posture Management) і CNAPP (Cloud-Native Application Protection Platform).

Ключове — телеметрія (логи, NetFlow/IPFIX, PCAP — packet capture, TLS-fingerprinting), її кореляція через SIEM і UEBA (User and Entity Behavior Analytics) та впровадження threat-hunting (полювання на загрози). Практичні рекомендації: defense-in-depth (багаторівневий захист), мікросегментація, інвестиції в телеметрію й автоматизацію, регулярні вправи (tabletop, red/blue/purple-team) та метрики ефективності — Time To Detect (TTD), Time To Contain (TTC) і відсоток покриття технік MITRE ATT&CK (Adversarial Tactics, Techniques and Common Knowledge).

3.1. МОДЕЛІ АТАКУВАЛЬНИКІВ І ЗАГРОЗ: ВІД ІНДИВІДУАЛЬНИХ ХАКЕРІВ ДО ДЕРЖАВНИХ АРТ-ГРУП

Організаційний сучасний ландшафт кіберзагроз демонструє диверсифікацію акторів, появу комерційних сервісів у підпільних ринках, а також швидку адаптацію тактик під нові технології (хмара, мобільні платформи, штучний інтелект). Модель класифікації атакувальників допомагає організаціям пріоритизувати контрзаходи за мотивом, можливостями та потенційним впливом. Нижче наведено розширені профілі основних категорій атакувальників із додаванням сучасної інформації й практичних рекомендацій.

Модель «індивідуальний хакер» або «хобіст» (часто жартівливо — script kiddie) описує одноосібного автора, що має обмежені технічні ресурси й зазвичай використовує готові скрипти та інструменти з відкритих джерел. Мотиви таких акторів здебільшого немотивовані комерційно — це прагнення самоствердження, цікавість або демонстрація навичок; їхні атаки

зазвичай хаотичні, непередбачувані й націлені на легкі цілі (невимушені веб-форми, погано захищені сервера). Для захисту від цих загроз ефективні базові заходи: регулярний патч-менеджмент (управління виправленнями), мінімальні привілеї, захист від SQL/XSS-ін'єкцій та базова конфігурація мережеских екранів — адже більшість атак експлуатують очевидні вразливості. Сьогодні навіть «хобісти» отримують доступ до складніших інструментів через Telegram-канали й даркнет-форуми, тому організації мають приділяти увагу автоматичній оцінці вразливостей і базовій орієнтації користувачів на кібергігієну; також корисно відпрацьовувати реакцію на масові скриптові сканування та брутфорс-атаки.



Рис. 3.1. Ілюстрація індивідуального хакера (script kiddie)
Джерело: PNGArts

Наступний клас — «хактивісти» (hacktivists), тобто актори, які діють з ідеологічних або політичних мотивів. Вони можуть організовувати DDoS-атаки (Distributed Denial of Service — розподілене відмовлення в обслуговуванні), публікацію викрадених даних або defacement (підміну) веб-ресурсів з метою привернути увагу. Хактивісти іноді демонструють достатній рівень технічності для використання експлойтів і скриптів-ботнетів, але рідко мають ресурсів для тривалої спрямованої розвідки.



Рис. 3.2. Ілюстрація «хактивістів» (hacktivists)
Джерело: PNGArts

Контрзаходи тут включають масштабовану DDoS-мітігацію, моніторинг аномалій у трафіку (NDR — Network Detection and Response), чіткі процеси публічних комунікацій і оперативні playbook-и реагування на інформаційні

інциденти. За останні роки хактивістські кампанії стали більш «гібридними»: вони поєднують публічну інформаційну кампанію з технічними діями (витоки, DDoS) і іноді кооперуються з кіберзлочинцями заради ресурсу або можливостей — тому треба поєднувати технічні мітри та PR-готовність в одному плані реагування.



Рис. 3.3. Життєвий цикл атаки хактивістів (Hactivism) / Attack Lifecycle of Hactivism

Джерело: <https://doubleoctopus.com/security-wiki/threats-and-tools/advanced-persistent-threat/>

«Кіберзлочинці» — дуже різноманітна категорія: від одиничних шахраїв до організованих кримінальних угруповань, що працюють за бізнес-моделлю Ransomware-as-a-Service (RaaS — «вимагачі як сервіс»). Мотив — фінансова вигода; методи — фішинг, розсилки з шкідливими вкладеннями, brute-force проти RDP/SSH, експлуатація вразливостей, латеральний рух і ексфільтрація. Ці групи інвестують у стабільну інфраструктуру (C2 — Command and Control сервера), мають модульну архітектуру шкідливого ПЗ і часто застосовують поліморфізм та анти-форензик. Сьогодні зростає роль посередників — initial access brokers (IAB), які продають доступи в мережі жертв, а RaaS-екосистема дозволяє швидко масштабувати кампанії; одночасно спостерігається загострення тренду ransomware-кампаній із високими вимогами та подвійним шантажем, що робить необхідним поєднання резервного копіювання, EDR/SIEM/NDR-моніторингу та оперативного IR-процесу.



Рис. 3.4. Ілюстрація кіберзлочинців

Джерело: PNGArts

Існує окрема суттєва категорія — «інсайдери» (insider threats): співробітники або підрядники, які навмисно або випадково створюють ризики. Інсайдери можуть бути привілейованими адміністраторами, незадоволеними працівниками або просто користувачами, що випадково викликають компрометацію (наприклад, клікнули на фішинг). Виявлення інсайдерських загроз спирається на пов'язану поведінкову аналітику (UEBA — User and Entity Behavior Analytics), контроль доступу за принципом найменших привілеїв (least privilege), розподіл обов'язків (segregation of duties) і процеси управління життєвим циклом доступу (provisioning/deprovisioning). Сучасні виклики включають постійний контрактинг зовнішніх підрядників і роботу в хмарі, тому важливо застосовувати розширений контроль сесій привілейованих користувачів (PAM — Privileged Access Management), детальне логування доступів до хмарних сервісів і регулярні огляди прав доступу, а також інтегрувати UEBA-сигнали у SOAR-процеси для автоматичної реакції на підозрілу поведінку.



Рис. 3.5. Ілюстрація категорій інсайдерських загроз — від навмисних та змовних дій до випадкових (ненавмисних) помилок і ризиків, пов'язаних зі сторонніми підрядниками

Джерело: PNGArts

На вершині складності — «державні» або «підтримані державою» АРТ-групи. Це довготривалі, ресурсно-насичені актори з високою

технічною компетенцією, чіткими цілями (розвідка, саботаж, політичний вплив) і здатністю вести цілеспрямовану розвідку та багатофазні операції. АРТ роблять глибоку рекогносцировку, використовують zero-day експлойти, supply-chain compromise (компрометацію ланцюга постачання), підроблені цифрові підписи й гіпервізорні або прошивкові техніки для персистентності. Їхня ефективність обумовлена синергією технічних навичок, ресурсів і розвідувальних цілей; прикладом масштабного supply-chain інциденту є атака на платформу управління оновленнями, що дозволила інжектувати шкідливий код у легітимні релізи й скомпрометувати тисячі клієнтів. Захист від АРТ вимагає загостреного фокусу на перевірці ланцюгів постачання, вимірюванні цілісності завантажувача/прошивок і тісному обміні threat-intelligence з державними CERT/CSIRT.

Є також «сервісні» моделі загроз: кіберзлочинні ринки пропонують послуги — від оренди ботнетів і продажу доступу до корпоративних мереж до послуг «чорного» реверс-інжинірингу; це робить загрози більш доступними і прискорює еволюцію інструментарію. Захист проти таких «послуг» значною мірою тотожний захисту від організованої злочинності, але вимагає додаткового фокусу на швидкому виявленні інфраструктури C2, DGA-патернів і розгортанні sinkhole/технік для перехоплення доменів. Паралельно з цим спостерігається зростання використання штучного інтелекту й автоматизації атак (для генерації фішингових текстів, підбору паролів, автоматичної обробки даних), що ускладнює оцінку масштабу загроз і вимагає від оборонців застосування власних AI-засобів для аналізу й відбору пріоритетів. Використання AI змінює швидкість і персоналізацію атак, тому defense-команди мають інвестувати в інструменти, які автоматично корелюють великі обсяги телеметрії й підвищують точність детекцій.

Коли моделюється загроза, важливо описувати її трьома вимірами: мотив (motivation), можливості (capabilities) й масштаб/напрямок (intent & targeting). Така класифікація допомагає пріоритизувати контрзаходи: низькомотивовані/низькоздатні актори вимагають базової гігієни; економічно мотивовані та організовані — детекційної інфраструктури, резервних копій і правового планування; АРТ — інтеграції threat-intelligence, апаратної телеметрії і співпраці з держструктурами. Практична мапа — зіставлення профілю актора з MITRE ATT&CK і Cyber Kill Chain — дозволяє визначити найбільш імовірні TTPs і побудувати coverage-matrix для пріоритизації detection engineering. Окрім технічних мап, сучасні програми захисту повинні враховувати фактори постачальницького ризику,

ризиків третьої сторони й юридичні обмеження обміну даними між країнами.

Атрибуція (визначення винуватця) залишається складною через проксі-інфраструктуру, використання крадених інструментів і мультифакторність операцій; тому практичний підхід — не «встановити винного» як першочергову мету, а швидко локалізувати й нейтралізувати інфраструктуру атаки та відновити бізнес-процеси. Операційно це означає: інтеграцію EDR, NDR, SIEM, SOAR і threat-intelligence; відпрацьовані IR-playbook-и; регулярні red/blue/purple-team вправи; і політики зберігання та тестування резервних копій. Сучасні кейси показують, що швидке блокування IOCs і автоматизовані containment-playbook-и значно знижують негативний вплив незалежно від остаточної атрибуції, тому інвестиції в оперативну автоматизацію та якісну телеметрію дають найшвидший ефект.

Короткі практичні рекомендації наприкінці: важливо забезпечити базову кібергігієну (патчі, MFA, least-privilege, резервні копії), розгорнути EDR/SIEM/NDR і SOAR для автоматизації реакцій, інтегрувати threat-intelligence і відпрацювати IR-процедури, а для захисту від найпотужніших APT інвестувати у апаратні механізми цілісності платформи (TPM, Secure Boot), пам'яттеву форензику та галузеву координацію. Регулярно оновлювати coverage-matrix на базі MITRE ATT&CK, тестувати правила детекції на історичних логах і проводити purple-team вправи для скорочення TTD/TTC. І нарешті — документувати всі рішення, метрики та навчання: це перетворює разові заходи в системну здатність організації протистояти еволюційним загрозам.

3.2. АТАКИ НА МЕРЕЖЕВОМУ РІВНІ: DDoS, ARP-SPOOFING, IP-SPOOFING

Атаки на мережевому рівні — це широке сімейство технік, спрямованих на порушення нормальної комунікації між вузлами або на встановлення несанкціонованого контролю над потоком даних. Серед найпоширеніших категорій — DDoS (Distributed Denial of Service — розподілене відмовлення в обслуговуванні), ARP-spoofing (атака підміни записів у протоколі Address Resolution Protocol) і IP-spoofing (фальсифікація полів джерельної IP-адреси в пакетах Інтернет-протоколу). Вони відрізняються цілями і методами: DDoS прагне вивести послугу з ладу через перевантаження, ARP-spoofing реалізує man-in-the-middle (людина-посередник) в локальній мережі, а IP-spoofing використовується як для маскування походження, так і як складова рефлекторних/ампліфікаційних

атак. До сучасних особливостей додається активне використання орендованих ботнетів (BaaS), автоматизованих бот-ферм для L7-автоботу та комбінація мережових і прикладних векторів у «гібридних» DDoS-кампаніях.

Інструментарій для проведення атак на мережевому (L3/L4) рівні

Інструментарій для проведення атак на мережевому (L3/L4) рівні охоплює широкий спектр засобів і підходів, які відрізняються за цілями, інтенсивністю та методами генерації трафіку. Концептуально його можна розглядати як набір компонентів для синтезу великого обсягу мережових пакетів, керування потоками та організації розподілених джерел — від простих генераторів одиничних потоків до складних систем, що координують множинні вузли й використовують механізми підробки адрес та віддзеркалення. На рівні мережових протоколів застосовуються методи, спрямовані на заповнення пропускну здатності каналів (volumetric), виснаження станових ресурсів стеку протоколів (state-exhaustion) та виклик некоректної роботи мережевого обладнання через аномальний розподіл пакетів. Характерні властивості такого інструментарію — можливість масштабування (вертикального й горизонтального), гнучкість у формуванні шаблонів трафіку, здатність маскувати походження джерел та адаптуватись до захисних заходів через зміни патернів трафіку в реальному часі.

З точки зору оператора або дослідника загроз важливо виділяти кілька логічних підсистем у такому інструментарії: генерація трафіку (модуль, який формує пакети з заданими ознаками: протоколами, розмірами, інтервалами), механізми координації (сервіси управління й синхронізації розподілених джерел), проксі/маскувальні шари (щоб ускладнити трасування походження), та модулі автоматичної адаптації (які змінюють патерни відповідно до реакції цілі). Такі підсистеми створюють фундамент для мультивекторних сценаріїв, коли одночасно застосовуються кілька векторів навантаження: великі обсяги UDP-/ICMP-пакетів, потік недоконаних TCP-з'єднань, а також перевантаження певних портів або служб. Особливу загрозу становлять методи, що використовують відкриті інтернет-сервіси як рефлектори/ампліфікатори: у таких сценаріях одна коротка вимога приводить до значного відповіді з боку стороннього сервера, що посилює загальний трафік проти цілі.

Аналіз ризиків і наслідків від застосування мережевого інструментарію для DDoS-атак підкреслює, що навіть якщо первинною метою є тимчасове блокування доступності, побічні ефекти часто виходять за рамки технічних втрат: падіння SLA, фінансові збитки, зростання витрат на послуги очищення трафіку, репутаційні ризики, ускладнення

розслідувань через використання маскувальних шарів і, у деяких випадках, ескалація атак в політичному або кримінальному контексті. З огляду на це, оцінка загрози повинна включати не лише технічну характеристику потенційного трафіку (bps, rps), а й ідентифікацію критичних точок інфраструктури, вразливих сервісів (наприклад, незахищених UDP-серверів, відкритих NTP/DNS), а також потенційні канали, через які атака може бути підсилена.

Для захисту важливо розуміти архітектурні наслідки цього інструментарію: ефективні контрзаходи спираються на багатошарову стратегію, що поєднує перехід критичних ресурсів на розподілені/георозподілені архітектури, співпрацю з провайдерами для фільтрації на краю мережі, застосування адаптивних політик rate-limiting та використання спеціалізованих центрів «очищення» трафіку. При цьому системи виявлення повинні поєднувати класичні порогові метрики (bps, rps, gps) з поведінковим аналізом, який виділяє аномалії в часових рядах, розподілі джерел за ASN/географією і нетипових паттернах з'єднань. Саме поєднання цих підходів дозволяє отримувати контекстні сигнали, зменшуючи ризик помилкової фільтрації легітимного трафіку.

У дослідницькому та навчальному контексті важливо також відрізняти шкідливі інструменти від легітимних засобів навантажувального тестування: останні створені для безпечної перевірки стійкості сервісів і підтримуються юридично (нагромажуюче тестування за згодою власника), містять механізми контролю й звітності і є корисними для підготовки інфраструктури до реальних загроз. Рекомендації для наукового аналізу мережевого інструментарію включають дотримання етичних і правових норм, проведення експериментів лише у контрольованих середовищах або з явною письмовою згодою, а також фокус на вимірюванні наслідків (MTTD, MTTR, peak bandwidth/pps, вплив на бізнес-метрики) та розробці адаптивних захисних політик на основі отриманих даних.

DDoS-атаки (Distributed Denial of Service attacks) - розподілені атаки відмови в обслуговуванні

DDoS-атаки бувають багаторівневими — від L3/L4 (мережевий/транспортний рівні) до L7 (рівень застосунків). На рівні L3/L4 типовими є ICMP-фlood, UDP-флуди, TCP-SYN-flood, а також ампліфікаційні атаки через відкриті інтернет-сервіси (наприклад, DNS-, NTP-, CLDAP-ампліфікація), коли зловмисник підроблює source IP і спровоковує сервер-рефлектор відправити величезний об'єм трафіку жертві. L7-атаки (HTTP/HTTPS-флуди, slowloris тощо) намагаються виснажити ресурси конкретної веб-служби, відтворюючи «легітимні» запити або блокуючи пул з'єднань. Сьогодні дедалі частіше застосовують DDoS як

інструмент шантажу (RDoS — ransom DDoS), атакуючі комбінують TLS-шифровані запити й API-абуз для обходу простих фільтрів, а також використовують розподілені хмарні ресурси й «bot-as-a-service», що робить атаки більш масштабними й важчими для блокування без провайдерської участі.

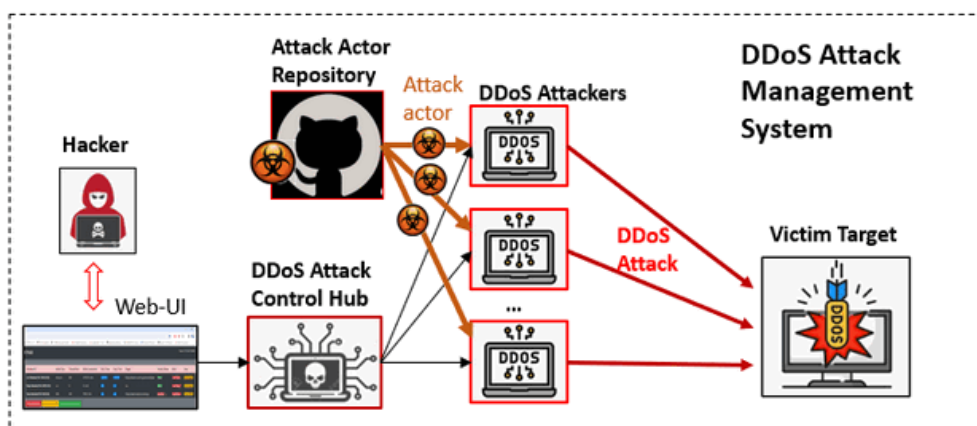


Рис. 3.6. Архітектура системи керування DDoS-атаками (DDoS Attack Management System)

Джерело: [1-9, 11-26, 32-48]

Етапи реалізації DDoS-атаки

Етапи реалізації DDoS-атаки сприймаються як послідовний життєвий цикл, у якому технічні, організаційні та мережеві компоненти взаємодіють для досягнення масштабного порушення доступності цільового сервісу.

На початковому етапі проводиться розвідка — визначення вразливостей інфраструктури, критичних точок доступу (edge-точки, API, балансувальники), пропускної здатності та поведінки при пікових навантаженнях; цей етап також включає збирання даних про провайдерів, топологію мережі та можливі точки фільтрації.

Далі йде фаза підготовки ресурсів атаки: створення або оренда ботнету, підготовка інструментів для генерації трафіку (включаючи техніки віддзеркалення й підсилення), налаштування маршрутів і налаштувань клієнтського трафіку для обходу базових фільтрів; для віддзеркалювальних/ампліфікаційних атак це також передбачає ідентифікацію публічних сервісів, які можуть послужити ретрансляторами. Виконання атаки охоплює синхронізоване спрямування великого обсягу трафіку на ціль із різноманітних джерел або на багаточастотні (multi-vector) вектори — обсягові (volumetric), протокольні (state-exhaustion) та прикладні (application-layer) — з метою перевантаження пропускної здатності, ресурсів стеку TCP/IP або логіки обробки запитів на прикладному рівні.

Підтримка атаки (sustainment) може включати періодичні хвилі (burst), адаптацію патернів у відповідь на застосовані захисні контрзаходи та використання обфускації джерел, а завершальна фаза передбачає відступ і спроби приховати сліди, одночасно посилюючи вторинні ефекти (наприклад, вимикання захисних механізмів цілі) — на останньому етапі ініціюється збір доказів для форензики й оцінка впливу на бізнес-процеси.

Інструментарій мережевого рівня, що використовується в DDoS-кампаніях

Інструментарій мережевого рівня, що використовується в DDoS-кампаніях, по суті формується з набору функціональних підсистем, кожна з яких має свої характерні властивості та наслідки для систем захисту. Перша підсистема — генерація трафіку — відповідає за синтез мережевих пакетів із заданими характеристиками: протоколом (ICMP/UDP/TCP тощо), розміром пакетів, інтервалом між пакетами та можливими варіантами фрагментації. Характеристика цієї підсистеми важлива для розуміння виду навантаження (об'ємне/pps-орієнтоване) і його впливу на різні елементи стеку (канальна пропускна здатність vs. CPU/контрольні структури стеку). Друга підсистема — інфраструктура розподілених джерел — забезпечує масштабування: здатність горизонтально нарощувати кількість джерел, координувати хвилі трафіку та зберігати синхронізацію між ними. З точки зору ризику, саме можливість масштабування визначає потенційний «підйом» пік-пропускної здатності і, відповідно, вимоги до заходів пом'якшення на рівні провайдера.

Третя важлива підсистема — механізми маскування і підробки походження пакетів — визначає складність трасування і блокувань. Характеристики цієї підсистеми включають здатність підміняти IP-поля (спуфінг), використовувати проксі/ретранслятори або багатоетапні ланцюги, а також інтеграцію з анонімними мережами. Четверта підсистема — механізми підсилення (рефлектори/ампліфікатори) — експлуатує протоколи й сервіси, що відповідають із значним коефіцієнтом підсилення, перетворюючи невеликі запити на великі відповіді; її характеристика включає вибір типів сервісів-рефлекторів, співвідношення запит/відповідь (ампліфікаційний множник) і ризик collateral damage для третіх сторін. П'ята підсистема — адаптаційні модулі — вміють змінювати патерни трафіку в реальному часі у відповідь на застосовані контрзаходи: коригування інтервалів, зміна розмірів пакетів, перемикання між векторами (наприклад, з volumetric → state-exhaustion) або перехід на зашифровані запити. Наявність таких адаптацій істотно ускладнює виявлення на основі статичних підписів і підвищує ймовірність тривалого впливу.

З погляду характеристик трафіку, інструментарій мережевого рівня може продукувати різні режими навантаження: високу пропускну здатність у біт/с (bps) для насичення каналів, велику кількість пакетів у секунду (pps) для виснаження обробних можливостей мережевого обладнання, або високий рівень нових/напіввідкритих з'єднань для виснаження станових таблиць на граничних пристроях. Кожен режим має свої метрики та індикатори: peak bandwidth, sustained throughput, pps, відсоток дефіцитних/втрачених пакетів, латентність сервісів та частота таймаутів. Крім того, характер розподілу джерел (географія, ASN, масштаб ботнету) — це окрема характеристика, яка впливає на вибір захисту: близький географічно, але скомпрометований пул має інші наслідки, ніж широко розподілена мережа з багатьох автономних систем.

Щодо протоколів — інструментарій може орієнтуватись на безпосередню роботу з IP/ICMP/UDP/TCP, а також використовувати особливості служб (NTP, DNS, SNMP тощо) для рефлексивних і ампліфікаційних сценаріїв. Характеристика тут — це не перелік уразливих сервісів, а розуміння, які властивості протоколу роблять його придатним до зловживання: наявність великого відгуку на малий запит, відсутність автентифікації або можливість розсилки від імені іншого адресата. Для захисту важливо моделювати ці властивості, аби визначити, де розумно застосовувати фільтрацію або обмеження.

Інструментарій часто включає також механізми управління й моніторингу атаки з боку оператора: канали команд/контролю, засоби телеметрії по джерелах і стану хвиль, механізми автоматичного перемикавання вектора. Характеристика таких засобів — ступінь централізації (централізований C2 vs. децентралізований) і здатність підтримувати тривалі операції з мінімальною видимістю. Для оборонців це означає, що реагування має мінімізувати час, коли атака має перевагу в координації; відповідно, важливі швидкі канали взаємодії з провайдерами й постачальниками послуг очищення.

З огляду на виявлення, характеристики інструментарію визначають набір детекторів і правил: інструментарій, орієнтований на високу пропускну здатність, створює явні порогові сигнали в bps/pps; інструментарій, що прагне прихованості, генерує малопомітні аномалії у часових рядах або імітує легітимні патерни, тому потребує поведінкового аналізу і кореляції метрик (джерела/ASN/географія/типи пакетів). Високий рівень спуфінгу або використання проксі знижує корисність простого блокування по IP, натомість підвищує цінність сигнатур на аномалії сесійних параметрів і аналізу рівня TCP/UDP-семантики.

Насамкінець, правова та етична характеристика інструментарію має бути зазначена окремо: інструменти, що можуть бути застосовані для навантажувального тестування, відрізняються від інструментів для злочинних дій саме наявністю контролю, дозволів і звітності. У дослідженнях і при розробці захисту слід суворо дотримуватися принципів: експерименти лише у контрольованому середовищі або на підставі письмового дозволу, ретельна анонімізація результатів, передача вразливостей відповідним адміністраторам і надання рекомендацій щодо пом'якшення. Оцінка інструментарію повинна завершуватися аналізом наслідків для бізнесу (SLA, фінансові втрати, репутаційні ризики) і рекомендаціями для багаторівневої стратегії захисту, що поєднує архітектурні (георозподіл, CDN), мережеві (фільтрація на периметрі, BGP-кооперація) і прикладні (кешування, rate-limiting, challenge-response) заходи.

Механізми запобігання та виявлення DDoS-атак

Механізми запобігання та виявлення DDoS-атак повинні реалізовуватися як багаторівнева стратегія захисту (defence-in-depth), що поєднує превентивні архітектурні рішення, мережеві та прикладні контрзаходи, інструменти автоматичного виявлення аномалій та регламентовані операційні процедури.

На архітектурному рівні ефективними є принципи розподіленої інфраструктури (Anycast-маршрутизація, георозподілені CDN і балансування навантаження), горизонтального масштабування сервісів та ізоляції критичних компонентів; ці підходи підвищують резилієнс магістральної пропускної здатності і дозволяють розосередити пікові хвилі трафіку. Мережеві контрзаходи включають кооперацію з провайдерами для фільтрації на периферії (ingress/egress filtering, BGP remote-triggered blackholing або sinkholing з координованими scrubber-центрами), застосування rate-limiting і traffic-shaping, а також налаштування ACL і інструментів для відсікання аномальних потоків за IP/порт/протоколом; для захисту від протокольних атак доцільні техніки жорсткої конфігурації стеку (наприклад, SYN-cookies, збільшення черг, таймаутів) та обмеження ресурсів на рівні мережевих елементів. На рівні прикладного шару необхідно застосовувати WAF і специфічні правила контекстної фільтрації для захисту ресурсомістких endpoint-ів (наприклад, API-шляхів), впроваджувати механізми кешування обслуговуваних сторінок і відповідей, перевести частину логіки на CDN, а також застосовувати adaptive throttling і challenge-response (CAPTCHA, proof-of-work) для підозрілих або аномальних сесій.

Системи виявлення і моніторингу мають поєднувати традиційні метрики (bps — біт/с, pps — пакетів/с, rps — запитів/с) із поведінковим аналізом та кореляцією подій у SIEM: побудова базових профілів трафіку, виявлення відхилень у часових рядах (анормальні хвилі), аналіз розподілу джерел трафіку (гео/ASN), а також швидке сповіщення при перевищенні заданих порогів. Анотомічні моделі та машинне навчання для DDoS-моніторингу дозволяють виділяти «нові» або «складні» вектори атак, що не підпадають під класичні фільтри; важливо, щоб системи виявлення давали контекстні сигнали, корельовані з бізнес-метриками (SLAs, latency, error-rates), аби уникнути надмірної фільтрації легітимного трафіку. Практика використання захисних сервісів (managed DDoS mitigation, scrubbing centers) є стандартом для критичних сервісів: при виявленні аномалії трафік направляється на очищення, де широкі сигнатури й евристики видаляють шкідливі потоки, а легітимні запити доставляються до цілі.

Операційні та процедурні елементи захисту включають підготовлені плейбуки реагування, координацію з провайдерами та регуляторами, регулярні навчання та drills (tabletop exercises) для команд SOC/NetOps, а також SLA-договірні механізми з постачальниками облачних/мережових послуг. Під час інциденту важливими є швидке переключення на заздалегідь узгоджені маршрути або scrubber-сервіси, активне throttle і блекхолінг на рівні підмереж за рішенням інцидент-лейдера, збереження telemetry та логів для форензики і пост-інцидентного аналізу. Після подолання атаки проводяться ретроспективний аналіз (post-mortem) із вимірюванням ключових показників (MTTD — mean time to detect, MTTR — mean time to recover, peak bandwidth/pps, бізнес-втрати) і корекцією архітектури та процесів: інвестиції у додаткову пропускну здатність, оптимізація правил фільтрації, оновлення захисних підписів і підвищення автоматизації реагування.

У підсумку, комплексний підхід, що поєднує архітектурну розподіленість, мережеву кооперацію з провайдерами, активні засоби фільтрації та автоматизоване виявлення аномалій, забезпечує найбільш стійку протидію DDoS-загрозам і дозволяє підтримувати прийнятний рівень доступності критичних інформаційних сервісів.

IP-spoofing (IP address spoofing) — підміна IP-адреси

IP-spoofing — це техніка, коли атакувальник підмінює поле source IP у заголовку пакета, щоб приховати своє походження або змусити відповідь на запит піти не до атакуючого, а до обраної жертви (рефлекторні атаки). IP-спуфінг лежить в основі багатьох ампліфікаційних операцій і ускладнює атрибуцію. Загалом, протидія цій загрозі базується на правильному фільтруванні трафіку на крайових маршрутизаторах: практики

ingress/egress filtering згідно з BCP38, а також включення unicast Reverse Path Forwarding (uRPF) на пристроях провайдерів і в мережі підприємства дозволяють блокувати пакети з несумісними або несподіваними source IP. Останні тренди показують, що перевірка адрес на краю мережі має доповнюватися аналітикою BGP (Flowspec, RPKI) і репутаційними списками джерел, а також автоматизованими алертами на аномалії геолокації й темпів зростання потоків, що прискорює реакцію на ампліфікаційні кампанії.

Етапи реалізації атаки

Етапи реалізації атаки (життєвий цикл IP-spoofing). IP-spoofing — це загальний прийом, коли атакувальник підмінює (фальсифікує) поле джерельної IP-адреси в заголовку мережевого пакета з метою маскування походження трафіку або створення неправдивого каналу взаємодії між джерелом і ціллю. Життєвий цикл реалізації такої загрози починається з розвідки мережевого середовища: вивчення топології, визначення діапазонів адрес цілей і проміжних вузлів, аналіз шляхів маршрутизації (AS-шляхи), а також ідентифікація сервісів, що можуть бути використані як ретранслятори (reflection/amplification vectors).

На підготовчому етапі атакувальник формує стратегію підміни адрес — вирішує, чи підмінюватиме окремі пакети конкретною адресою жертви (для обходу фільтрів), чи генеруватиме масовані пакети з випадковими або підбірними джерелами для ампліфікації ефекту; одночасно оцінюється необхідність використання віддзеркалюючих сервісів (наприклад, UDP-сервісів з високим коефіцієнтом посилення). Виконання полягає в генеруванні і відправці пакетів зі сфальсифікованими IP-заголовками відповідно до обраної стратегії: це може бути одинична операція для обходу механізмів фільтрації чи масштабована кампанія з метою виснаження ресурсів цілі або створення помилкового трафіку, який ускладнює ідентифікацію реального джерела. Паралельно з фазою виконання відбувається адаптація тактики у відповідь на захисні заходи (наприклад, зміна патернів джерел, частотності чи типів використаних сервісів) та спроби ускладнити трасування — наприклад, за допомогою мульти-AS маршрутів або багатоетапної ретрансляції.

Завершальна фаза включає операції з приховання слідів і аналізу ефективності (збір телеметрії, оцінка досягнутого впливу на доступність/цілісність мережі), а також, у випадку зл�якісних кампаній, заходи з підтримання контролю над ресурсами або їхнього продажу на чорному ринку. Важливо: технічні деталі реалізації IP-spoofing не повинні бути використані для протиправних дій — опис тут має концептуальний і

науковий характер, спрямований на розуміння загрози й розробку контрзаходів.

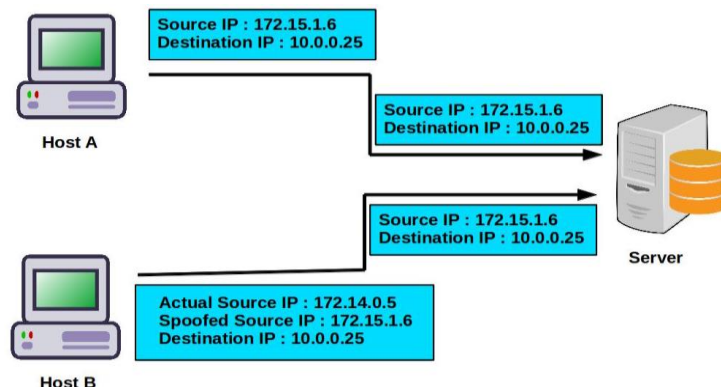


Рис. 3.7. IP-спуфінг — приклад підміни IP-адреси джерела (Source IP spoofing)

Джерело: [1-9, 11-26, 32-48]

Інструментарій мережевого рівня, що стосується підміни IP-адрес (IP-spoofing)

Інструментарій мережевого рівня, що опосередковано стосується підміни IP-адрес (IP-spoofing), слід розглядати не як набір конкретних програм, а як сукупність функціональних можливостей і архітектурних компонентів, які створюють умови для підміняння джерела пакета або імітації легітимного джерела. У центрі уваги такого інструментарію — можливість формувати мережеві пакети з контрольованими значеннями полів заголовків (зокрема IP-source), керувати інтервалами й обсягами трафіку, а також координувати множинні джерела для створення розподілених або віддзеркалювальних сценаріїв. Ключові характеристики: гнучкість у формуванні заголовків і семантики пакетів, здатність масштабувати чисельність джерел і хвиль, наявність механізмів маскуванню ланцюга передачі (перехід через проксі/ретранслятори), і можливість автоматичної зміни патернів відповідно до реакції захисних систем.

З практично-аналітичної точки зору важливо підкреслити, що IP-спуфінг сам по собі часто використовується як складова інших атак — наприклад, у віддзеркалювальних/ампліфікаційних сценаріях, де підмінена адреса цілі спонукає третю сторону відправити великий обсяг відповіді на жертву, або в спробах ускладнити трасування джерел при багатовекторних DDoS-кампаніях. Тому характеристики інструментарію потрібно описувати через вплив на показники трафіку: підвищені bps/pps на певні порти або сервіси, незвична дисперсія джерел (широке охоплення ASN/географій) або, навпаки, концентрація «фальшивих» джерел у межах певних діапазонів.

Також важлива поведінкова властивість — інструментарій, орієнтований на прихованість, породжує малопомітні аномалії в часових рядах і намагається імітувати легітимні шаблони сесій, що ускладнює виявлення традиційними пороговими детекторами.

Індикатори компрометації та сліди, які породжує активність із підміною адрес, мають переважно статистичний та кореляційний характер: непокєднуваність зворотного маршруту (asymmetric routing anomalies), високий відсоток ICMP-повідомлень «destination unreachable» або «time exceeded», невідповідність геолокації джерел очікуваному профілю клієнтів, а також підвищена частота аномалій у таблицях станів (наприклад, велика кількість напіввідкритих TCP-сесій із невідповідними ACK/SEQ). Для захисних систем важливо поєднувати ці сигнали з мережевою телеметрією (NetFlow/IPFIX), даними маршрутизації (BGP updates, AS-origins) і контекстною інформацією з SIEM, щоб відрізнити шкідливі патерни від легітимних сплесків трафіку.

З погляду запобігання і пом'якшення, характеристики інструментарію диктують набір пріоритетних контрзаходів: валідація джерела на межі мережі (фільтрація вихідного трафіку — egress filtering) і вхідна фільтрація (ingress), політики перевірки зворотного шляху (reverse-path checks), обмеження та сегментація сервісів, а також кооперація з провайдерами й реєстрами маршрутизації (RPKI/ROA) для зменшення можливостей підміни на рівні глобального маршрутування. Технічні заходи повинні поєднуватися з архітектурними — використання георозподілених CDN, балансування навантаження і scrubber-сервісів — оскільки інструментарій, здатний масштабуватися, робить локальні одноточкові рішення неефективними.

Нарешті, етичний і правовий контекст є невід'ємною частиною характеристики: будь-які експерименти або симуляції, пов'язані з підміною адрес, повинні виконуватись виключно у контрольованому середовищі або за явної письмової згоди власника інфраструктури; у межах наукових робіт доцільно описувати моделі та результати у знеособленому вигляді й надавати рекомендації щодо захисту, а не практичні інструкції щодо реалізації атак. Для аналітичного розділу в роботі рекомендовано завершити опис переліком метрик (bps, pps, ratio of spoofed flows, ASN dispersion, peak packet size distribution) і набором оперативних рекомендацій для SOC/NetOps: налаштування збірки NetFlow/IPFIX, тригерів у SIEM на аномальні профілі джерел, процес координації з провайдерами та регулярні вправи з реагування у лабораторних умовах.

Механізми запобігання та виявлення IP-spoofing

Механізми запобігання та виявлення IP-spoofing (комплексні контрзаходи). Запобігання і виявлення підміни IP-адрес вимагають поєднання мережесих політик, протоколів контролю маршрутизації, засобів фільтрації та аналітики трафіку. На межі мережі та в операційній практиці основним превентивним заходом є Source Address Validation — ingress/egress filtering відповідно до рекомендацій (BCP38/BCP84): провайдери і підприємницькі мережі повинні відкидати пакети з некоректними або неочікуваними джерельними адресами на вході та обмежувати вихідний трафік лише тими адресами, які дійсно належать абоненту.

Технічною реалізацією цього підходу є механізми uRPF (Unicast Reverse Path Forwarding) на маршрутизаторах, строгі ACL і політики egress-filtering; на рівні глобальної маршрутизації — застосування RPKI/ROA для валідації походження префіксів та автентифікації BGP-анонсів, що зменшує можливість маршрутної фальсифікації, яка часто супроводжує складні атаки із підміною адрес.

Для пом'якшення наслідків ампліфікаційних і віддзеркалювальних атак корисні мережесі практики hardening: закриття та фрагментація сервісів, що здатні слугувати підсилювачами; обмеження відповідей на несправжні запити (rate limiting на UDP-сервери, налаштування регуляризації протоколів), конфігурація DNS/NTP/Network-devices так, щоб вони не відповідали на запити з неперевічених джерел або не працювали у режимі open-resolver/monlist. Кооперація з провайдерами щодо BGP-фільтрації та швидкого реагування на виявлені джерела атак (RTBH — remote triggered blackholing, або перенаправлення трафіку на scrubber-центри) є практичним елементом реагування при масових інцидентах.

Методи виявлення включають аналіз потоків (NetFlow/sFlow/IPFIX), кореляцію телеметрії та поведінкову аналітику: аномалії у розподілі джерел (висока частка «нових» або випадкових source IP), незвичні патерни TTL, розбіжності у геолокації/AS для джерел, надмірна кількість одночасних сесій із різних адрес у межах одного підключення — усе це є індикаторами піддробленого походження. Технічно корисні техніки TTL-/hop-count-аналізу (перевірка очікуваного TTL для пакета від заданої підмережі), парування IP з MAC-адресами на ланцюговому (L2) рівні, ідентифікація несумісності між IP та власними маршрутними таблицями — ці підходи дозволяють виявляти невідповідності, притаманні spoofed-пакетам. Системи IDS/IPS і SIEM повинні корелювати індикатори з різних джерел: мережесих сенсорів, логів маршрутизаторів і серверів, щоб відокремити легітимний розподілений трафік від штучно згенерованого. Для трасування походження атак застосовують техніки traceback (packet marking, flow

tracing) і координацію між мережевими операторами; у практиці це часто вимагає міжоператорської співпраці та законодавчої підтримки.

Процедурні та організаційні заходи включають політики egress/ingress-фільтрації на рівні клієнта й провайдера, регулярні аудити конфігурацій маршрутизаторів і служб, закриття потенційно небезпечних сервісів, і плани координації з інтернет-провайдерами у разі атак (контакти CSIRT, обмін IoC). Висновок: ефективна протидія IP-spoofing вимагає системного підходу — поєднання мережевої фільтрації, маршрутизаційної безпеки (RPKI), поведінкової аналітики потоків та міжоператорської кооперації — що дозволяє як мінімізувати можливість підміни адрес, так і швидко виявляти та пом'якшувати наслідки таких інцидентів.

ARP-spoofing (ARP-cache poisoning) — підміна ARP-записів

ARP-spoofing (ARP-cache poisoning) — це атака локальної мережі (Layer 2), в якій зловмисник підробляє відповіді протоколу Address Resolution Protocol, спрямовуючи трафік відхилити або проходити через його вузол. В результаті відбувається перехоплення даних, модифікація трафіку або скидання сесій.

Наслідки — перехоплення облікових даних, сесій, впровадження шкідливих payload-ів або створення умов для подальшого lateral movement в мережі. Сучасні захисні практики також враховують аналогічні загрози в IPv6 (ND/RA-spoofing) і використовують механізми на кшталт RA-Guard, Secure Neighbor Discovery (SEND) де доступно, а також рішення SDN/NAC для динамічного блокування підозрілих портів і швидкого карантину підозрілих пристроїв.

Виявлення мережевих атак потребує багаторівневої телеметрії та кореляції: NetFlow/sFlow/IPFIX-збір для виявлення аномалій в обсягах і патернах трафіку, NDR (Network Detection and Response)-системи для глибшого аналізу протоколів і поведінки, а також інтеграція з SIEM (Security Information and Event Management) для кореляції сигналів від EDR/он-хост-логів і мережевих джерел.

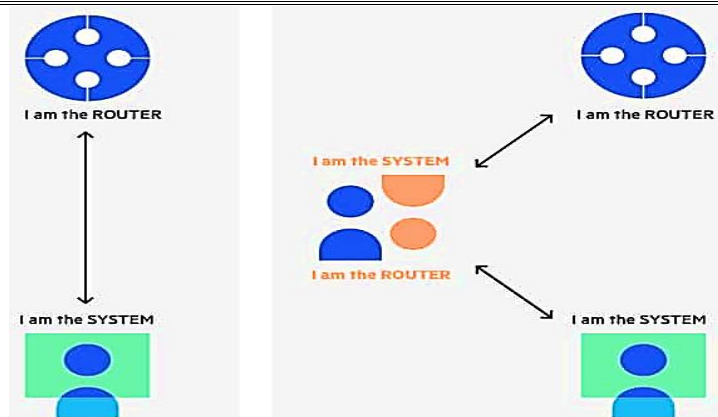


Рис. 3.8. ARP-poisoning — підміна MAC-записів та перехоплення трафіку
Джерело: [1-9, 11-26, 32-48]

Для DDoS важливими сигналами є різке зростання числа сесій, аномалії в розподілі портів/протоколів, високий відсоток fragmented packets або незвична частота SYN/ACK. Для ARP-spoofing індикаторами є часті зміни MAC→IP-відповідностей, дублювання MAC-адрес або непередбачувані gratuitous ARP-повідомлення. Сьогодні додатково застосовують TLS/HTTP fingerprinting (наприклад JA3/JA3S), поведінкові моделі на основі ML для виявлення «bot-like» патернів, а також кореляцію з threat-intelligence (IOC) і загальною телеметрією CDN/провайдерів для швидкої класифікації подій й автоматичного включення mitigation-процедур.

Етапи реалізації атаки

Етапи реалізації атаки (життєвий цикл ARP-spoofing / ARP-cache poisoning). ARP-spoofing (ARP-cache poisoning) — це мережевий прийом, що експлуатує відсутність автентифікації у протоколі ARP: зловмисник підмінює відповідь ARP або надсилає хибні gratuitous ARP-повідомлення, щоб змінити відповідність IP ↔ MAC у ARP-кеші інших хостів і тимчасово перенаправити локальний трафік через свій вузол. Життєвий цикл такої загрози у загальних рисах складається з послідовних етапів: розвідка, ідентифікація цілей, концептуальне впровадження підміни, експлуатація і пост-експлуатаційний аналіз.

На фазі розвідки зловмисник оцінює локальну топологію (які підмережі та сегменти доступні, які хости є критичними — шлюзи, сервери), збирає інформацію про MAC-таблиці комутаторів і виявляє потенційні «вузли-перехватчики» (наприклад, шлюз або сервер аутентифікації).

На етапі ідентифікації цілей досліджуються можливі жертви та їх поведінка (частота ARP-запитів, час життя кешу, наявність спостереження/моніторингу), оцінюється ризик виявлення і потенційна

користь (можливість перехоплення сесій, підміни трафіку, впровадження нападів MitM).

Фаза впровадження концептуальна: тут визначається модель впливу на ARP-кеші (тимчасове отруєння записів у вибіркових вузлах або масова підміна) та оцінюються можливості перехоплення, інтерпретації чи модифікації трафіку без викриття. Якщо ARP-підміна успішна, настає фаза експлуатації — перехоплення локального трафіку, сніффінг незашифрованих протоколів, ін'єкція або перенаправлення з метою фішингу/сесійного захоплення/сканування внутрішніх ресурсів; одночасно атакувальник оцінює сліди своєї діяльності (записи в логах, зміни MAC-таблиць комутаторів) і коригує тактику для зменшення виявлення.

Завершальна фаза передбачає приховування слідів, аналіз ефективності кампанії та документування «результатів» у разі кримінальних або етичних досліджень; у легальному і науковому контексті весь цикл має бути відтворений і задокументований лише в ізольованому середовищі або за письмовою згодою власника мережі.

Практика пом'якшення наслідків DDoS включає: договори з провайдерами на швидке reroute/blackholing, використання Anycast-архітектури і CDN (Content Delivery Network) для розподілу навантаження, підключення до DDoS-scrubbing-центрів, застосування rate limiting і connection throttling на краю мережі, а також застосування WAF (Web Application Firewall) для захисту L7-сервісів. Для локальних атак (ARP) та спуфінгу IP — комбінування технічних механізмів на інфраструктурному рівні (DAI, uRPF, BCP38), сегментація та шифрування трафіку. Додаткові сучасні контрзаходи включають bot-management платформи (challenge/response, fingerprinting), інтеграцію CDN і WAF з автоматичними rulesets, а також використання хмарних провайдерів безпеки з можливістю швидкого автоскейлу миттєвого фільтрування зловмисного трафіку.

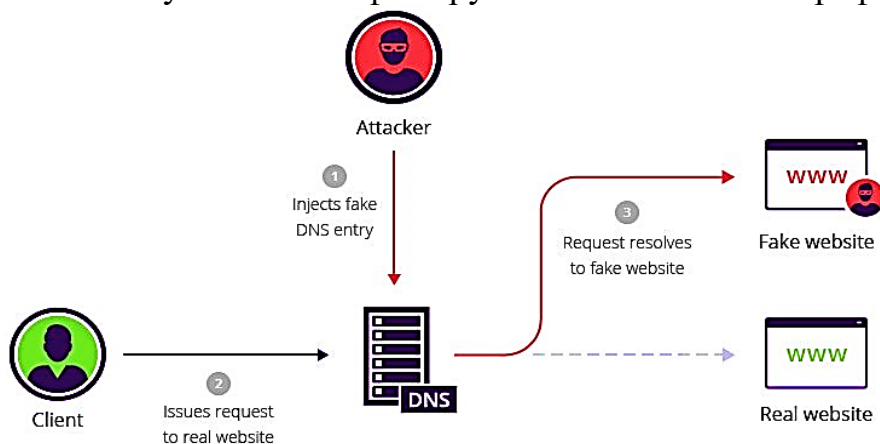


Рис. 3.9. Схема підміни DNS-записів (DNS-spoofing / DNS cache poisoning)
Джерело: [1-9, 11-26, 32-48]

Інструментарій мережевого рівня, що опосередковано стосується атаки ARP-spoofing

ARP-спуфінг (підміна ARP-записів) є локальною мережею-орієнтованою загрозою, тому «інструментарій», що опосередковано до неї відноситься, слід розглядати як сукупність функціональних можливостей й архітектурних умов, які полегшують або ускладнюють реалізацію сценаріїв підміни MAC↔IP-відповідностей, а не як перелік конкретних програм. На рівні характеристик та можливостей такий інструментарій включає (описано узагальнено й без практичної деталізації) можливості формування ARP-повідомлень із довільним вмістом заголовків, маніпулювання ARP-кешем на вузлах мережі, а також засоби перехоплення й аналізу локальної мережевої телеметрії для оцінки ефекту змінених ARP-відповідностей. Важливими властивостями є: гнучкість у зміні MAC↔IP-пар, здатність швидко поширювати «грубі» або «безшумні» (low-rate) оновлення ARP-кешів, можливість взаємодіяти з механізмами комутації й мостування (наприклад, перевіряти чи зберігаються MAC-відповідності у CAM-таблицях комутаторів) та наявність каналів команд-контролю для координації дій у межах локальної мережі.

З погляду аналітика загроз, корисно розрізняти дві ролі таких характеристик: «ініціювальні» (тобто що дозволяють вплинути на таблиці адрес) та «маскувальні/стійкі» (тобто що дозволяють зменшити виявлення змін). Ініціювальні можливості пов'язані з частотою й обсягом ARP-повідомлень та зі здатністю змінювати час життя записів у кеші; маскувальні — з можливістю імітувати коректну поведінку клієнтів (наприклад, відповіді лише за потреби, обмежені часові вікна) або використовувати одночасні патерни, що нагадують легітимні мережеві події. Характеристика «масштабованості» в локальних ARP-сценаріях проявляється в можливості впливати на значну підмережу шляхом спрямованих оновлень ARP або використання комбінації ARP-підмін з іншими техніками (наприклад, маніпуляціями з DHCP), що змінює загальний профіль трафіку й ускладнює просте фільтрування.

Індикатори, які породжує активність, пов'язана з підміною ARP-записів, мають переважно кореляційний і статистичний характер. Для SOC/NetOps варто відстежувати такі сигнали: часті або раптові зміни MAC-адреси для однієї й тієї ж IP-адреси (ARP-flapping), множинні різні IP-адреси, що відповідають одній й тій самій MAC-адресі, підвищення частоти ARP-запитів/відповідей у короткому інтервалі (spikes in ARP traffic), невідповідність ARP-таблиць між суміжними комутаторами, а також зростання числа ICMP-повідомлень або TCP/UDP-помилки, що свідчить

про проблеми маршрутування на рівні L2/L3. Додаткові контекстні сигнали — незвична поведінка служб (наприклад, знижений трафік від очікуваного сервісу або зникнення ARP-записів під час піків), невідповідність даних DHCP та ARP (IP, зарезервовані через DHCP, але з «чужими» MAC), і розбіжності між NetFlow/IPFIX даними й локальними ARP-записами. Ці індикатори ефективні в комплексі; кожен поодиноці може бути легітимним явищем, але сукупність сигналів вказує на ризик підміни.

Форензичні сліди та телеметрія, корисні для післяінцидентного аналізу, включають збережені дампи ARP-таблиць на мережевих пристроях і кінцевих вузлах, журнали комутаторів (CAM-таблиці), дані DHCP-серверів, записи NetFlow/IPFIX та системних журналів ОС клієнтів. Корисною є кореляція часових міток між подіями: наприклад, поява нового ARP-запису в таблиці плюс одночасне зростання помилок ARP чи ARP-запитів у суміжних сегментах. Важливо, щоб логування ARP-подій і мережевої телеметрії було налаштоване заздалегідь, оскільки пізніше відновити стан мережі складно.

Захисні та превентивні характеристики інструментарію диктують набір рекомендованих контрзаходів, які слід реалізувати на рівні архітектури й конфігурації мережі. Архітектурні підходи (географічна/логічна сегментація VLAN, мінімізація широкомовних доменів, використання гостьових/IoT-сегментів) зменшують площу потенційного впливу. Конфігураційні заходи на комутаторах і точках доступу — це застосування функцій запобігання на L2: обмеження навчання MAC-адрес (port-security), фіксовані ARP-записи там, де це припустимо, активація механізмів контролю прив'язки IP↔MAC (наприклад, Dynamic ARP Inspection у суміші з DHCP Snooping у корпоративних комутаторах) — всі ці рішення змінюють «поведінкові властивості» мережі так, щоб спуфінг-активність ставала помітною або неможливою. На рівні доступу рекомендується впровадження 802.1X, чітка сегментація прав доступу й політик, а також застосування процедур управління життєвими циклами адрес (статичні прив'язки для критичних вузлів, контроль за DHCP-резерваціями).

Моніторинг і виявлення — ще один важливий блок захисту: SIEM має отримувати тригерні події для аномалій ARP (ARP-flapping, різкі стрибки ARP-трафіку, несумісність DHCP↔ARP), NetFlow/IPFIX має корелювати потоки з локальними ARP-таблицями, а мережеві сенсори/IDS повинні мати сигнатури й евристики, що виявляють аномальні патерни L2. Важливо враховувати баланс між чутливістю й рівнем хибних спрацьовувань: занадто суворі правила можуть перервати легітимну мережеву динаміку.

Роль людини у циклі реагування — підтвердження сигналу через кілька незалежних джерел — залишається критичною.

Нарешті, правовий й етичний контекст: будь-яке тестування поведінки ARP-інфраструктури чи імітація загроз повинно проводитись лише у контрольованому середовищі або після отримання письмових дозволів від власників інфраструктури. У наукових роботах слід описувати методи моніторингу, моделі й результати в обезособленому вигляді, а не надавати практичні рекомендації щодо реалізації атак. Аналіз інструментарію має завершуватись переліком метрик для оцінки ризику та ефективності захисту (частота ARP-змін для IP, кількість унікальних MAC для IP за годину, ratio of gratuitous ARP, ARP request rate per port, CAM table churn rate) та набором практичних рекомендацій для SOC/NetOps: впровадити моніторинг ARP у SIEM, регулярні перевірки CAM-таблиць, налаштування DAI/DHCP-snooping там, де це підтримується, навчальні drills для команд і процедурну координацію з мережевими провайдерами у разі інциденту.

Механізми запобігання та виявлення ARP-spoofing

Механізми запобігання та виявлення ARP-spoofing (комплексні контрзаходи). Ефективна протидія підміні ARP базується на поєднанні мережових конфігураційних політик, функцій комутаторів, засобів моніторингу й гігієни протоколів додатку (шифрування), тобто на принципі defence-in-depth. На рівні комутації та контролю доступу доцільно застосувати такі технічні заходи: жорстку сегментацію мережі (VLAN) і обмеження широкомовних доменів ARP; використання функцій мережових пристроїв — static ARP-entries для критичних серверів і шлюзів там, де це керовано і масштабовано, налаштування Port Security для фіксації очікуваних MAC-адрес на портах комутаторів, а також включення механізмів DHCP Snooping і Dynamic ARP Inspection (DAI) на сумісних комутаторах, причому DAI опирається на binding-таблиці DHCP для валідації ARP-відповідей. На рівні хостів корисні локальні політики кешування й фіксації (наприклад, контроль над TTL/timeout для ARP-записів у межах прийнятних значень) та застосування захищених протоколів доступу (SSH/HTTPS/TLS) — шифрування трафіку робить перехоплення ARP-перенаправленого трафіку менш корисним.

Механізми виявлення повинні поєднувати прості хости-орієнтовані індикатори і мережовий аналіз: сповіщення про незвичну зміну локальної ARP-таблиці (раптова зміна MAC для IP- шлюзу), надходження частих gratuitous ARP або аномальної кількості ARP-відповідей, «flapping» MAC-адрес у CAM-таблиці комутатора, неспівпадіння MAC↔IP у записах NetFlow/ARP, а також розбіжності TTL/hop-count у суміжних потоках — все

це є індикаторами можливого отруєння кешу. Інструменти моніторингу (arpwatch, спеціалізовані модулі IDS/IPS, вбудовані засоби DAI) можуть генерувати алерти на ці індикатори; SIEM-кореляція логів комутаторів, dhcp-серверів і кінцевих вузлів підвищує точність виявлення і дозволяє відфільтровувати помилкові спрацьовування. Для просунутого виявлення застосовують пасивний аналіз трафіку на мережевих сенсорах (SPAN/mirror, TAP) і порівняння ARP-відповідей з очікуваною базою відповідностей (IP $\leftrightarrow$ MAC), а також періодичні active-checks у контрольованих сценаріях для перевірки консистентності таблиць.

Операційні процедури і організаційні заходи доповнюють технічні контрзаходи: регулярні аудити конфігурацій мережевих пристроїв, контроль життєвого циклу DHCP-binding і адресних планів, політики щодо додавання/переміщення пристроїв у мережі (верифікація MAC), а також навчання операторів і користувачів виявляти симптоми MITM-поведінки (SSL/TLS-warnings, незвичні сайти при доступі до відомих ресурсів). У разі інциденту плани реагування повинні передбачати швидку ізоляцію підмережі або порту, очищення ARP-кешів на хостах і комутаторах, відновлення статичних прив'язок для критичних ресурсів, збір мережевих дамів і CAM/ARP-таблиць для форензики, а також ротацію зламаних ключів/сертифікатів, якщо була компрометація секретів. Як додатковий захід варто впровадити політику «шифрування за замовчуванням» (TLS і HTTP Strict Transport Security, certificate pinning там, де це доцільно), оскільки навіть при успішному ARP-митм шифровані сесії значно ускладнюють ексфільтрацію корисної інформації.

Науково-технічна оцінка ефективності контрзаходів для ARP-spoofing може будуватися через метрики виявлення (TPR/FPR для арп-алертів), час виявлення (MTTD) і час відновлення (MTTR), а також через формалізацію зниження вразливості мережевої поверхні (напр., зменшення домену broadcast/ARP на кожному сегменті). У сукупності, поєднання управління доступом на рівні комутаторів, інспекції і валідації ARP-повідомлень (DAI), сегментації мережі та повсюдного шифрування трафіку забезпечує стійкий захист від ARP-cache poisoning та мінімізує потенційні наслідки у випадку спроб внутрішньої компрометації.

Реагування на мережеві інциденти повинно бути регламентованим: швидке виявлення та триаж, ідентифікація векторів і обсягу (L3/L4 vs L7), застосування відпрацьованих механізмів mitigation (edge-filters, rate limiting, blackholing, scrubbing), паралельний збір доказів (PCAP, flow-логи, BGP-route logs) для подальшої форензики і after-action review. Постінцидентний аналіз має визначити слабкі місця архітектури (однопунктові точки відмови, недостатнє резервування каналів, відсутність фільтрації на кордоні) і

оновити playbook-и, правила на обладнанні та контракти із зовнішніми постачальниками послуг. Сучасні процедури також передбачають автоматизацію через SOAR-плейбуки для швидкого виконання mitigation-кроків, регулярні tabletop-вправи з провайдерами та включення правових/PR-алгоритмів для координації повідомлень і можливих судових дій. Важливо зберігати можливість масштабного збирання телеметрії та її довготривалого архівування — це критично для розробки нових детекцій і співпраці в рамках ISAC/CERT.

На завершення — ефективний захист від мережеских атак комбінує превентивні заходи (фільтри BCP38, uRPF, DAI, сегментація мережі), постійний моніторинг (NetFlow/NDR/SIEM), готові канали для миттєвого mitigation (ISP, scrubbing) і навчання операційних команд. Тільки такий багатосаровий і координаційний підхід дозволяє одночасно зменшувати ризик успішної реалізації IP-спуфінгу і ARP-підміни та швидко нейтралізувати масштабні DDoS-кампанії, що загрожують доступності критичних сервісів. У сучасному контексті до цього набору потрібно додати елементи zero-trust, micro-segmentation, SASE/secure SD-WAN та конвергентні XDR-рішення, які дозволяють корелювати сигнал на рівнях endpoint, мережі та хмари. Розвиток автоматизованих, оркестрованих процесів реагування та регулярний обмін розвідданими роблять захист більш адаптивним і скорочують вікно вразливості перед новими мультивекторними атаками.

3.3. АТАКИ НА ТРАНСПОРТНОМУ РІВНІ: TCP HIJACKING, UDP FLOODING, SYN FLOOD

Атаки на транспортному рівні орієнтовані на порушення коректності роботи транспортних протоколів (передусім TCP і UDP) або на захоплення існуючих сесій; їхня мета — позбавити сервіс доступності, перехопити управління сесією або виснажити ресурси стеку протоколів. У сучасних мережах ці атаки залишаються одними з найпрагматичніших: вони використовують фундаментальні особливості TCP (стан з'єднання, трьохступеневе встановлення сеансу) та UDP (безстанність) і часто комбінуються з мережевими техніками (спуфінг, ампліфікація) для підвищення ефективності. Нижче — детальний огляд трьох ключових категорій: TCP hijacking, UDP flooding і SYN flood — з поясненням механіки, ознак виявлення та практичних заходів пом'якшення. Сучасний ландшафт також включає нові стекові технології (QUIC/HTTP/3 над UDP, TLS 1.3) і розподілені хмарні сервіси, що змінюють як вектори атак, так і

доступні механізми захисту — це потрібно враховувати при проектуванні детекції й mitigation-планів.

TCP hijacking (захоплення TCP-сесії)

TCP hijacking (захоплення TCP-сесії) — це техніка, коли атакувальник перехоплює існуючу TCP-сесію або вводить у неї свої пакети, видаючи їх за пакети легітимного учасника. Існують різні варіанти: пасивний перехват (sniffing) дозволяє зчитати незашифровані дані; активний захоплюючий інжект (packet injection) — припускає підробку пакетів з відповідними номерами послідовності (sequence numbers) і, можливо, завершення легітимної сесії через RST. Раніше часто використовувалися атаки на основі прогнозування TCP-sequence, проте сучасні стекові реалізації та захист (рандомізація начального ISN, TLS) значно ускладнили цей підхід. Практичними індикаторами TCP-hijack є несподівана зміна поведінки сесії (раптові команди/відповіді), поява TCP RST з незвичних джерел, а також розбіжності між NetFlow/conntrack-логами і реальними потоками.

Сьогодні більшість ефективних захоплень пов'язані не стільки з класичним прогнозуванням послідовностей, скільки з крадіжкою сесійних токенів (cookie, OAuth-токени) через фішинг, XSS або незашифровані канали — тому захист від client-side атак має таке саме значення, як мережеві контрзаходи. Поява QUIC і HTTP/3 зміщує акцент: оскільки QUIC інтегрує TLS поверх UDP, традиційні мережеві інструменти для аналізу трафіку стають менш ефективними, отже детекція має орієнтуватися на поведінкові ознаки та мета-дані сесій. Рекомендовано впроваджувати TLS 1.3, ECH/ESNI для захисту SNI, використовувати secure/HttpOnly та SameSite для cookie, а також застосовувати MFA і частіше ротацію токенів для зниження ризику sidejacking. У мережевих операціях корисні eBPF/XDP-фільтри для швидкого виявлення і блокування підроблених пакетів та аномалій у порядку байтів, а також інтеграція з UEBA для виявлення незвичної поведінки користувача або сесії.

Етапи реалізації атаки TCP-hijacking

TCP-hijacking — це клас атак, спрямованих на захоплення або підміну існуючої TCP-сесії між двома кінцевими вузлами з метою прослуховування трафіку, ін'єкції пакетів або встановлення контрольованого каналу замість легітимного учасника. Життєвий цикл такої атаки починається з розвідки: збір інформації про мережеву топологію, активні сесії та відкриті TCP-з'єднання (IP-адреси, порти, часові характеристики), а також визначення можливих перехоплювальних позицій (на шляху трафіку — L2/L3-пристрої, проміжні проксі, або у вигляді локального MitM через ARP-spoofing чи compromise).

На етапі підготовки атакувальник визначає метод захоплення: пасивне відстеження і підслуховування для отримання поточних sequence/ack-значень (де це технічно можливо), активне втручання як-от TCP reset-інжекція з метою розірвати сесію і швидко вставити себе, або MITM-позиція для послідовної ін'єкції і корекції потоку.

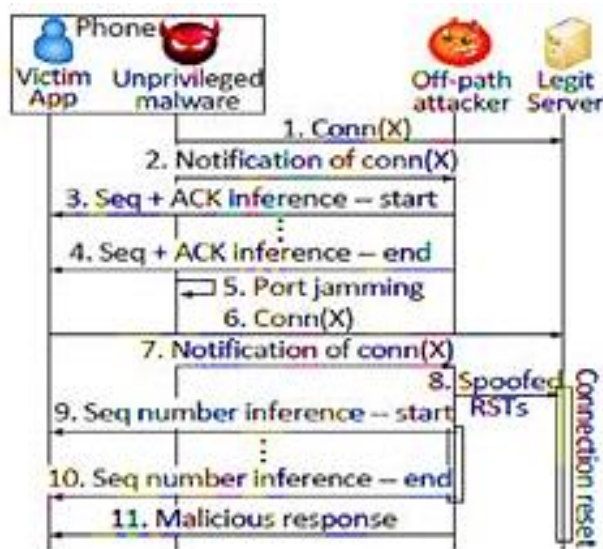


Рис. 3.10. Активне захоплення TCP-сесії (Active TCP Hijacking): послідовність дій атакуючого — від перехоплення/спостереження за номером послідовності до ін'єкції шкідливих пакетів і завершення легітимної сесії

Джерело: [1-9, 11-26, 32-48, 60-65]

Фаза верифікації і реалізації передбачає спроби синхронізуватися зі станом секвенційних номерів, відправити пакет з коректними параметрами та відповідними таймінгами або викликати reestablish таким чином, щоб жертва продовжувала взаємодію з атакуючим; важливо, що деталі реалізації залежать від конкретного стеку TCP, наявності шифрування та мережних контролів. Після успішного встановлення контролю йде фаза експлуатації: ексфільтрація даних, ін'єкція команд у прикладні протоколи, перенаправлення з'єднань чи ескалація доступу на базі перехоплених сесій; завершальною є фаза приховування слідів і аналізу ефективності, включно з оцінкою логів, залишків трафіку й можливістю подальшого використання сесії. Усі тестування і відладка методик мають проводитися лише у контрольованих лабораторних умовах або за письмовою згодою власника інфраструктури.

Інструментарій транспортного рівня, що опосередковано стосується загроз типу TCP-hijacking

Інструментарій транспортного рівня, що опосередковано стосується загроз типу TCP-hijacking, слід розглядати як набір функціональних можливостей і умов мережевої та хостової інфраструктури, які роблять цю загрозу можливою або ускладнюють її виявлення. У центрі уваги — можливість впливати на стан TCP-з'єднання або на його сприйняття учасниками (клієнтом і сервером) без авторизованого доступу до обох кінців сесії. Тому «інструментарій» тут — це не конкретні програми, а поєднання характеристик: здатність спостерігати трафік на рівні пакетів (пакетна телеметрія, mirror/span, pcap), модулі для кореляції сесійних параметрів (NetFlow/IPFIX, conntrack), засоби маніпуляції мережею L2/L3 (що можуть змінювати маршрут або видимість вузлів), а також наявність слабких місць у стеку TCP або у прикладному протоколі (відсутність криптографічної захищеності/аутентифікації, передача стану сесії у відкритому вигляді).

Ключові властивості такого інструментарію включають: телеметрична спроможність — можливість збирати і зберігати пакетні дампи та агреговані потоки для подальшого відтворення сесій; кореляційні механізми — здатність зіставляти потоки з логами додатків, таймстемпами і подіями ОС; видимість шляху трафіку — наскільки мережеві елементи (маршрутизатори, балансувальники, проксі) змінюють або ховають реальні кінцеві точки сесії; стійкість/динаміка сесій — як довго живуть сесії, чи використовуються короткоживучі токени/ідентифікатори, чи застосовується TLS з перевіркою сертифікатів; автоматизація реагування — можливість автоматично розривати підозрілі сесії, переключати трафік або інструменти очищення.

Індикатори, які може породити активність, пов'язана з захопленням TCP-сесії або спробами її підробити, мають мережево-пакетний і прикладний характер. До корисних сигналів належать: поява аномальних RST/FIN-пакетів у критичний момент сесії; повторювані duplicate ACK або аномально часті retransmission-події; раптові розриви сесій, після яких з'являються нові сесії з тією ж парою адрес/портів але з несумісною поведінкою; розбіжності між логами аутентифікації додатку і мережевими сесіями (наприклад, запис у журналі застосунку від імені користувача без відповідного TCP-потoku в мережесих дампах); невідповідності таймінгів і розмірів вікон TCP для відомих клієнтів. У поєднанні з поведінковим аналізом (ненормальні послідовності команд у прикладному протоколі, незвичні послідовності запит/відповідь) ці сигнали вказують на ризик компрометації сесії. Важливо: жоден індивідуальний індикатор не є однозначним — кореляція між джерелами даних підвищує достовірність виявлення.

Для форензики цінною телеметрією є: повні пакетні дампи (pcap) з часовими мітками з різних точок мережі (точки входу/виходу, балансувальники, кінцеві вузли), NetFlow/IPFIX записи з конектативними метриками (bps, pps, bytes per flow, flow duration), логи ОС і додатків (аутентифікація, сесійні токени, час останньої активності), журнали NAT/проксі та записи балансувальників. Кореляція часових міток між цими джерелами дозволяє реконструювати «момент втручання» та оцінити, чи була здійснена несанкціонована транзакція в рамках сесії. Для відновлення необхідно зберігати метадані і дампи ще до моменту реагування (retention policy), і мати процеси для правової опрацювання доказів.

Архітектурні та прикладні контрзаходи, які знижують привабливість векторів захоплення сесій, роблять їх значно менш ефективними: (а) застосування криптографічного захисту каналу (TLS/DTLS) з коректною перевіркою сертифікатів і строгими налаштуваннями шифровальних стеків; (б) впровадження аутентифікації й цілісності на рівні прикладання (користувацькі токени, HMAC-підписи звернень, короткоживучі маркери) — це забезпечує, що скомпрометована мережева сесія сама по собі не достатня для авторизованих дій; (в) використання механізмів захисту TCP-рівня де це можливо (наприклад, TCP Authentication Option / TCP-АО у критичних середовищах) як додаткового шару; (г) обмеження видимості мережі та сегментація: ізоляція критичних сервісів за допомогою VPN, внутрішніх проксі і reverse proxies, де додаткові контрольні точки виконують перевірки; (д) на рівні інфраструктури — налаштування балансувальників і WAF для виявлення аномальної прикладної поведінки, а також застосування time-based і behavioural rate limiting, яке враховує не тільки кількість пакетів, а й семантику сесій.

Операційні практики мають не менш важливе значення: регулярна перевірка конфігурацій (NAT/proxy/balancer), аудит сертифікатів і політик TLS, моніторинг цілісності сесійних токенів, тренування SOC у виявленні нетипових TCP-аномалій, та наявність playbook-ів для реагування, які включають збір поліційно-прийнятних доказів. Також корисною є практика «захищеного відтворення» — тестування відновлення інцидентів у контрольованому середовищі (lab drills) без активних дій у продакшні.

Механізми запобігання та виявлення TCP-hijacking

Надійна протидія TCP-hijacking базується на поєднанні шифрування каналів, жорсткої автентифікації на прикладному рівні, мережевих політик і поведінкового моніторингу. На рівні каналу і протоколу найбільш ефективним заходом є застосування криптографічного захисту (TLS/SSL, SSH), який робить непотрібними знання фактичних sequence-номерів для корисної експлуатації — шифрування і автентифікація кінцевих точок

усувають можливість невидимої ін'єкції пакетів у сесію. Додатково сучасні стек-реалізації TCP повинні впроваджувати захист від прогнозованих ISN (ініціалізаційних номерів) відповідно до RFC (наприклад, рандомізація ISN, захист від повторного відтворення). На рівні прикладного шару рекомендовано прив'язувати сесію до додаткових атрибутів (token-binding, client fingerprinting, контроль IP/UA-комбінацій), вимагати повторної автентифікації при критичних операціях та використовувати короткоживучі маркери сесії з автоматичною ротацією.

Мережеві й інфраструктурні контрзаходи включають захист від MitM-позицій: на L2 рівні — запобігання ARP-спуфінгу (DAI, Port Security, статичні ARP для критичних вузлів), на рівні маршрутизації — валідацію BGP і політик фільтрації, а також використання VPN/IPsec між сегментами, що зменшує ризик локального перехоплення. Для захисту від TCP reset-інжекцій і подібних атак застосовуються механізми фільтрації та rate-limit на граничних роутерах, а також налаштування стеку ОС (відмова від довільної реакції на RST від непередбачуваних джерел). Для внутрішніх сервісів корисне використання reverse-проху і load-balancer'ів із TLS-термінацією і вузькими політиками доступу, що мінімізує поверхню для прямих TCP-з'єднань.

Моніторинг і виявлення мають поєднувати низку ознак: аномалії у послідовності пакетів (стрибки sequence/ack), появи одночасних з'єднань з однаковими 4-tuple (srcIP, srcPort, dstIP, dstPort) з різних мереж, несподівані RST/FIN пакети, неочікувані retransmissions або зміни в характері трафіку прикладного рівня (відмінність у паттернах запит/відповідь). Інструменти для цього — NetFlow/IPFIX, пакетні сенсори (SPAN/TAP), IDS/IPS з правилами виявлення аномалій TCP-потoku і SIEM-кореляція логів (наприклад, поява нетипових source IP у сесії, повторні логіни або одночасні сесії з одного облікового запису). Поведінкові алгоритми і ML-моделі можуть виявляти відхилення від профілів нормальної сесії (час між запитом і відповіддю, обсяг переданих даних) і генерувати тригер на детальнішу перевірку чи тимчасову ізоляцію сесії.

Операційні та процесні заходи включають жорсткі політики управління сесіями і ключами (короткий TTL для сесійних токенів, ротація ключів), регулярне оновлення стеків та патчі ОС і мережевого обладнання, відмова від небезпечних відкритих протоколів (telnet, ftp без шифрування), а також тестування на проникнення і навчання операційних команд щодо індикаторів захоплення сесій. У разі підозри на hijacking процедури реагування мають передбачати негайну інвалідацію сесійних токенів, ротацію критичних облікових даних, збір мережевих дамів і логів для форензики, а також нотифікацію за встановленими політиками. Як частина

оцінки ефективності захисту доцільно використовувати метрики MTTD/MTTR, кількість виявлених аномалій на 1000 сесій і зниження ймовірності успіху атаки P_{success} у моделі ризику; комбінований defence-in-depth підхід (шарове шифрування, автентифікація, мережеві політики, моніторинг і процедури реагування) забезпечує найвищу стійкість проти TCP-hijacking.

UDP flooding (User Datagram Protocol flooding) - атака затоплення на основі протоколу UDP

UDP flooding експлуатує безстанну природу UDP: відправка великого обсягу UDP-пакетів на ціль або на послуги, що відповідають на запити (DNS, NTP, Memcached), призводить до вичерпування каналу, процесорних ресурсів або заповнення таблиць NAT/ACL. Часто UDP-флуд поєднують з ампліфікацією: атакувальник надсилає маленькі запити до відкритих UDP-серверів з підробленим source IP жертви, і ті у відповідь генерують суттєво більший об'єм трафіку до жертви. Виявлення UDP-флуду базується на телеметрії: різке зростання UDP-пакетів за хвилину/секунду, висока ентропія портів/дестинацій або велика кількість необроблених ICMP-повідомлень «destination unreachable». Пом'якшення — багаторівневе: фільтрація та rate-limiting на крайових маршрутизаторах, закриття/жорстка конфігурація відомих UDP-рефлекторів (DNS/NT P) у власній інфраструктурі, застосування anycast/CDN для розподілу навантаження і підключення до scrubbing-сервісів або провайдера з DDoS-mitigation.

Наразі багато класичних ампліфікаторів було частково нейтралізовано (закриті або патчені NTP/Chargen/SSDP), але з'являються нові послуги й пристрої IoT як потенційні рефлектори, тому інвентаризація відкритих UDP-сервісів лишається критичною. Сучасні системи виявлення використовують ML-моделі для розпізнавання аномалій у паттернах UDP-трафіку та автоматичну класифікацію типів атак, що дозволяє більш точно застосовувати rate-limits або blackholing. Для внутрішніх сервісів радять вводити response-rate-limiting (RRL) на DNS-серверах, відключати небезпечні функції (як-от memcached UDP) і застосовувати ACL/geo-fencing для обмеження джерел. Кооперація з провайдерами і застосування BGP Flowspec або RTBH дозволяють швидко відсікати великі поточкові хвилі до scrubbing-центрів, мінімізуючи вплив на бізнес-трафік.

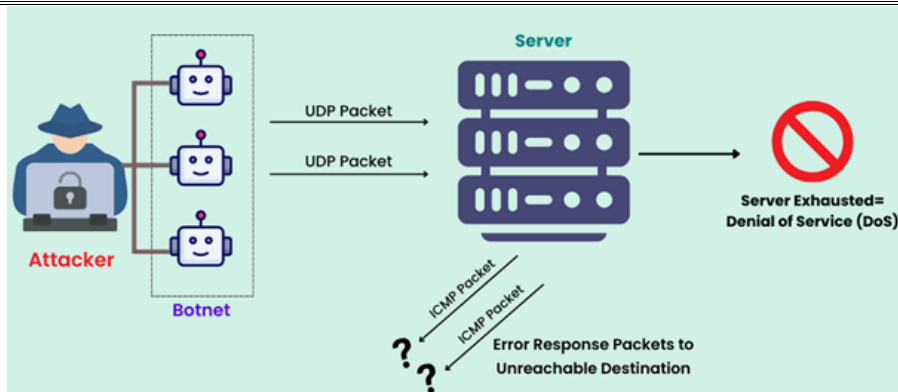


Рис. 3.11. Схема UDP-атаки (UDP flood): ботнет надсилає велику кількість UDP-пакетів на сервіс жертви або на відкриті UDP-рефлектори, що призводить до виснаження мережесих/системних ресурсів і відмови в обслуговуванні

Джерело: [1-9, 11-26, 32-48, 60-65]

Етапи реалізації UDP-атаки

DP-flood — це різновид DDoS-кампаній, що базується на генерації великого обсягу UDP-пакетів з метою виснаження пропускної здатності каналу, ресурсів мережесих елементів або обробних потужностей цільового сервісу; через безстанність протоколу UDP такі атаки легко масштабуються, часто супроводжуються фальсифікацією джерельних адрес і можуть використовувати техніки віддзеркалення/ампліфікації (DNS, NTP, memcached тощо).

Життєвий цикл реалізації UDP-flood має кілька логічних етапів. На стадії розвідки зловмисник досліджує топологію цілі, доступні UDP-сервіси й точки, які найчутливіші до UDP-навантаження (наприклад, DNS-рекурсори, VoIP-шлюзи, ігрові сервери), а також вивчає провайдерську інфраструктуру та можливості маршрутизаційної фільтрації.

На підготовчій фазі акумулюються ресурси атаки: будуються або орендуються ботнети, підбираються інструменти генерації трафіку, ідентифікуються open-resolver/reflector-сервіси для ампліфікації, налаштовуються патерни пакетів (розмір, порти, payload) для обходу простих фільтрів.

Фаза виконання характеризується масованою відправкою UDP-пакетів на ціль за обраною стратегією (volumetric flood, application-layer UDP flood, fragmented UDP), з одночасною адаптацією тактики у відповідь на виявлені захисні дії (зміна портів, швидкості, використання нових ретрансляторів). Підтримка (sustainment) атаки може включати періодичні піки, змішування векторів (multi-vector) і використання spoofed-джерел для ускладнення трасування; завершальна фаза передбачає зусилля з

приховання слідів, аналіз досягнутого впливу та підготовку подальших кампаній або «відступ». Весь цикл зазвичай супроводжується оцінкою ризику й можливостей контраоперацій, а також підрахунком потенційного економічного збитку для цілі.

Інструментарій транспортного рівня, що опосередковано стосується UDP-flooding

Інструментарій транспортного рівня, що опосередковано стосується UDP-flooding, доцільно розглядати не як перелік конкретних програм, а як сукупність функціональних здатностей і архітектурних умов, які визначають можливість генерації, масштабування, маскування та адаптації великих потоків безстанного UDP-трафіку. На рівні характеристик та можливостей такий інструментарій включає: можливість синтезувати потоки UDP-пакетів з контрольованими параметрами (розмір пакета, інтервал відправлення, цільові порти/діапазони); здатність підробляти джерело (що робить можливим віддзеркалення/ампліфікацію); інфраструктуру розподілених джерел або проксі-ланцюгів для горизонтального масштабування; та механізми автоматичного перемикавання між векторами (наприклад, перехід від простого UDP-флуду до використання рефлекторів із великим коефіцієнтом відповіді). Важливо, що ключовою властивістю є безстанна природа UDP — відсутність «розмовної» семантики робить цей рівень придатним для стрімкого нарощення пакето/байт-інтенсивності без необхідності підтримувати сесію.

З погляду впливу на інфраструктуру, характеристиками інструментарію є здатність створювати режими навантаження з різними метриками: високий sustained bps (насичення каналу), велика rps-інтенсивність (виснаження обробних циклів мережевого обладнання), а також патерни з великою ентропією портів і дестинацій (що ускладнює селективне фільтрування). Особлива увага в аналітиці приділяється ампліфікаційним сценаріям: властивості «рефлектора» (коефіцієнт amplification, розмір відповіді/запиту, відсутність автентифікації) визначають ефективність підсилення малих запитів у великі відповіді, що посилює вимоги до заходів пом'якшення на рівні провайдера.

Детекція та телеметрія сильно залежать від цих характеристик. Типові індикатори, які породжує активність такого інструментарію: раптове й значне зростання UDP-пакетів за секунду/хвилину відносно базового профілю; аномально висока частка ICMP «destination unreachable» у відповідь на спрямовані UDP-потоки; збільшення частоти короткоживучих потоків і великий розкид цільових портів; підвищена дисперсія джерел за ASN/географією або, навпаки, концентровані хвилі з невеликого набору AS. Кореляція NetFlow/IPFIX (bps, pps, bytes per flow, flow duration) з пакетними

дампами та системними лічильниками на граничних пристроях дає найбільш надійну картину — особливо якщо аналізуємо співвідношення запит/відповідь для відомих UDP-сервісів (ratio of responses to requests) і швидкі зміни в розподілі портів.

Пом'якшувальні заходи мають бути багаторівневими і орієнтованими на мінімізацію collateral damage: стратегічні (архітектурні) — георозподіл трафіку через anycast/CDN, сегментація сервісів і відведення критичних API за заздалегідь узгоджені шляхи; мережеві — фільтрація та rate-limiting на крайових маршрутизаторах, BGP-кооперація з провайдерами (RTBH, Flowspec або направлення на scrubbing-центри) і закриття/жорстка конфігурація служб, здатних до ампліфікації; прикладні — response-rate-limiting на DNS та інших UDP-серверів, відключення ризикових UDP-функцій у сервісах (коли це прийнятно), застосування ACL/geo-fencing. Окремо слід підкреслити важливість інвентаризації відкритих UDP-сервісів і регулярного патчування/нейтралізації відомих ампліфікаторів, а також контролю пристроїв IoT, які можуть служити як рефлектори.

Сучасні системи виявлення дедалі частіше застосовують моделі машинного навчання для виявлення нетипових шаблонів UDP-трафіку (анормальна ентропія портів, нетипова тривалість потоків, різка зміна співвідношення запит/відповідь). Проте ML-підходи повинні доповнюватися класичними пороговими тригерами і кореляцією з контекстними даними (події у SIEM, BGP-updates, інформація від провайдера), щоб знизити ризик хибних спрацювань. Практичною операційною рекомендацією є налаштування тригерів на відхилення від базового середнього (наприклад, значне збільшення bps/pps у відношенні до нормального профілю; у документації наукових робіт зазвичай дають приклади «>2–3× від середнього» як ілюстрацію, але конкретні пороги мають підганятися під свою інфраструктуру).

Для SOC/NetOps корисно фокусуватися на наборі метрик і сигнатур, що покривають транспортний аспект UDP-загроз: peak bandwidth, sustained throughput, packets per second, flow duration distribution, port entropy, ratio of ICMP errors to UDP flows, distribution by ASN/geolocation, та частка трафіку, що корелюється з відомими відкритими сервісами (DNS/NTP/memcached тощо). Операційні процедури повинні включати механізми швидкої координації з провайдерами, попередньо погоджені playbook-и для направлення трафіку в scrubber-центри, а також drills для відпрацювання сценаріїв, щоб зменшити MTTD і MTTR при хвилі UDP-навантаження.

Еволюція загроз підкреслює, що навіть після часткового закриття класичних ампліфікаторів нові сервіси та пристрої IoT можуть стати джерелом рефлексії, тому стратегія захисту має включати як технічні

контрзаходи, так і регулярну інвентаризацію зовнішніх і внутрішніх UDP-сервісів, політики мінімізації відкритих UDP-портів, а також програму співпраці з провайдерами щодо швидкого застосування BGP-рішень у разі масштабних хвиль.

Механізми запобігання та виявлення UDP-атак

Протидія UDP-флудінгу повинна будуватися мультишарово та поєднувати мережеві, інфраструктурні й процесні заходи. На периферії мережі пріоритетними є превентивні мережеві політики: ingress/egress-фільтрація і BCP38 для блокування трафіку з підробленими джерельними адресами, застосування uRPF на граничних маршрутизаторах, жорсткі ACL і rate-limiting на портах, а також BGP-фільтрація і RPKI для зменшення можливостей маршрутної підміни.

Архітектурно рекомендовано розподілену інфраструктуру (Anycast, CDN, георозподілені балансувальники), яка розсіює трафік і знижує навантаження на одиничні точки відмови; для критичних сервісів стандартною практикою є підключення до managed DDoS mitigation і scrubbing-центрів, які можуть перенаправляти й очищувати трафік при піку атаки.

На рівні мережевих пристроїв застосовуються техніки traffic-shaping, token-bucket policing, adaptive rate-limiting, фільтрація по портах/протоколам та обмеження на розмір і частоту фрагментованих UDP-пакетів; важливо також закривати/жорстко конфігурувати UDP-сервіси, що можуть слугувати рефлєкторами (DNS open-resolvers, NTP with monlist, memcached) або застосовувати відповідні обмеження відповіді (response rate limiting на DNS). Для захисту прикладного рівня доцільні кешування, edge-rate limiting і challenge-response (CAPTCHA, proof-of-work) для відокремлення автоматизованих потоків від легітимних сесій (хоча реалізувати це для UDP-протоколів складніше, іноді застосовують UDP-level puzzles або перехід на TCP/QUIC для критичних операцій).

Механізми виявлення повинні поєднувати швидкі мережеві метрики та поведінкову аналітику: моніторинг bps/pps/gps у реальному часі, аналіз NetFlow/sFlow/IPFIX для виявлення аномалій у джерелах (великі хвилі з нових/розпорошених source IP), ентропійний аналіз полів (source IP, dest port), виявлення значних змін у розподілі TTL/hop-count і частоті фрагментацій. SIEM-кореляція, сигналізація при перевищенні порогів і ML-моделі для часових рядів допомагають відрізнити сплески легітимного трафіку (наприклад, flash crowd) від зловмисних хвиль. Пакетні сенсори (SPAN/TAP), аналіз підозрілих payload-шаблонів і виявлення однотипних UDP-запитів з великою швидкістю — ключові інструменти раннього виявлення. Для мереж із великим ризиком ампліфікації корисно моніторити

відповіді рефлекторів і виявляти нетипово великі обсяги відповідей на малий запитний трафік — індикатор використання відкритих рефлекторів.

Операційні процедури реагування та відновлення включають чітко прописані плейбуки: швидке перекриття трафіку (blackholing/RTBH) на рівні провайдера, перенаправлення на scrubber-центри, динамічне застосування фільтрів і throttle-політик, а також збереження телеметрії (psar, flow-записи) для форензики. Координація з transit-провайдерами і CSIRT є критичною для масштабних атак, як і попереднє укладення договорів з mitigator-постачальниками. Після стихання атаки слід провести post-mortem з аналізом MTTD/MTTR, пікових показників (peak bps/pps), економічних втрат і корекцією архітектури (додавання Anycast, збільшення пропускної здатності, оптимізація правил фільтрації).

У підсумку, стійкість до UDP-флудінгу забезпечується поєднанням мережевої гігієни (BCP38, uRPF), архітектурної розподіленості (Anycast/CDN), превентивної конфігурації сервісів (закриття open-resolvers, RRL) та активного моніторингу з готовими процедурами реагування і кооперацією з провайдерами.

SYN flood (TCP SYN flood) - атака з перевантаження SYN-запитами

SYN flood — класичний приклад атаки на TCP-handshake: атакувальник масово надсилає SYN-пакети до сервера і не завершує handshake (не надсилає ACK), внаслідок чого сервер витрачає пам'ять і дескриптори на «half-open» з'єднання, що може вичерпати таблиці очікування і заблокувати легітимні підключення. Сучасні варіанти використовують спуфінг source IP для уникнення трейсбека; інша технологія — низькошвидкісні SYN-потoki, які поступово перевантажують стек. Для захисту існує набір добре відпрацьованих методів: SYN-cookies (технологія, що не створює стану до отримання ACK), SYN-cache механізми, збільшення розміру backlog, та налаштування TCP-таймаутів. На мережевому обладнанні корисні rate-limit по SYN, blackholing певних потоків, а також застосування спеціалізованих DDoS-приладів або провайдерських scrubbing-рішень.

Серед сучасних підходів — використання XDP/eBPF на краєвих маршрутизаторах для відкидання аномальних SYN-потоків ще на L3/L4 до передачі в стек ядра, а також SYN-проху в розподілених edge-точках хмари, що приховують реальні сервери від масових SYN-відповідей. Хмарні провайдери (AWS Shield, Azure DDoS Protection, Cloudflare Spectrum) пропонують автоматичне виявлення і mitigation, що значно скорочує time-to-contain для масштабних атак. Також зростає роль behavioral baselines: системи, що відстежують нормальні SYN/ACK-патерни для конкретного

додатку, можуть миттєво виявляти аномалії і застосовувати специфічні фільтри. Крім того, варто тестувати екстремальні сценарії (DDoS tabletop та load tests) у безпечному середовищі, щоб впевнитися, що fallback-механізми і резервні канали спрацювають коректно.

Усі три категорії вимагають не лише технічних фільтрів, а й правильної організаційної підготовки. До початку інциденту варто мати: базові пороги спрацювань у NDR/SIEM (SYN/second, UDP pps, аномалії сесій), контракти з провайдерами по mitigation, тестовані playbook-и (алгоритми blackholing, scrubbing, rate limiting), і процедури збору доказів (PCAP, NetFlow, BGP-route logs). Під час інциденту оперативні дії — верифікація джерела/масштабу, активація upstream-фільтрів, тимчасове застосування rate limiting та паралельний збір телеметрії для форензики. Після ліквідації інциденту обов'язково виконати post-mortem: розгорнути статистику, оцінити ефективність заходів (TTD/TTC), оновити thresholds і playbook-и, а також впровадити заходи довготривалої стійкості (anycast, CDN, hardening серверних стеків).

Сучасні готові практики включають попередні угоди про швидкий доступ до scrubbing-центрів і тестування BGP Flowspec або RTBH у співпраці з ISP, щоб уникнути бюрократичних затримок під час реальної атаки. Також важливо інтегрувати threat-intelligence про інфраструктуру атакуючих (IOC, DGA-патерни) у SIEM та автоматично підвантажувати оновлені сигнатури у мережеві фільтри. Юридичні та PR-процедури повинні бути погоджені заздалегідь: хто інформує клієнтів, регуляторів або партнерів і якою буде комунікаційна стратегія. І нарешті, регулярні вправи (red-team/blue-team, DDoS tabletop) значно підвищують готовність операційних команд і скорочують час від виявлення до containment.

Етапи реалізації атаки

SYN-flood є спеціалізованою формою DDoS-атак, що спрямована на виснаження таблиць стану або черг очікування прийому в стеку TCP на цільових серверах і мережевому обладнанні шляхом масованої ініціації TCP-handshake (SYN-пакети) без завершення рукопотискання. Життєвий цикл такої кампанії традиційно включає послідовні етапи: розвідка, підготовка ресурсів, фаза виконання, підтримка атаки і постінцидентний аналіз.

На стадії розвідки аналізується топологія цілі, налаштування TCP-стеку, розміри backlog-черг, поведінка серверів при пікових навантаженнях і наявність мережевих елементів (load-balancer, NAT), що впливають на стійкість — також перевіряється, чи дозволяють пристрої приймати та формувати відповіді на SYN від підроблених джерел. Підготовча фаза передбачає акумуляцію ресурсів (ботнети або інструменти генерації

трафіку), вибір патерну SYN (інтенсивність, розподіл по портах/адресах, частота) та опрацювання способів обходу базових фільтрів (спуфінг джерельних адрес, мультивекторні хвилі).

У фазі виконання відбувається масована відправка SYN-пакетів на ціль із заданим патерном: пакети можуть бути з фабрикованими source-IP, мати варіації TCP-опцій або приходити у вигляді швидких сплесків чи тривалих хвиль для підтримки навантаження; мета — заповнити доступні ресурси (memory, backlog, TCB) і викликати відмову або деградацію сервісу.

Фаза підтримки включає адаптацію тактики відповідно до вжитих контрзаходів (зміна швидкості, джерел, портів), щоб зберегти тиск на ціль, а завершальна фаза — аналіз результатів, приховування слідів та підготовка наступних кампаній або звіту в разі наукового/законного тестування. Всі випробування й моделювання повинні виконуватись лише в контрольованих середовищах або за письмовим дозволом власника інфраструктури; технічні деталі не повинні використовуватися з шкідливою метою.

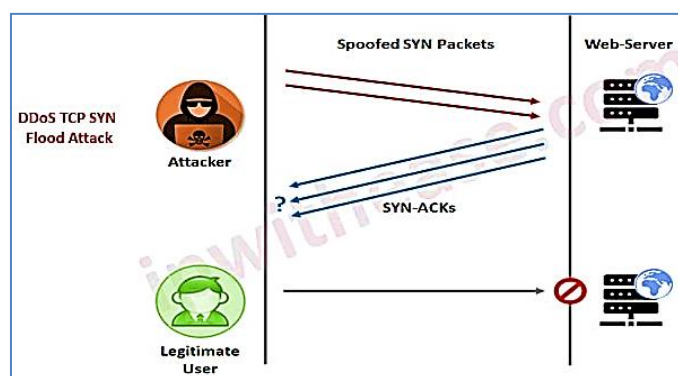


Рис. 3.12. Схема TCP SYN-Flood (DDoS) — підробка SYN-пакетів і створення «half-open» з'єднань

Джерело: [1-9, 11-26, 32-48, 60-65]

Інструментарій транспортного рівня, релевантний до сценаріїв SYN-flood

Інструментарій транспортного рівня, релевантний до сценаріїв SYN-flood, варто розглядати як сукупність функціональних можливостей і архітектурних умов, які визначають потенціал для створення масового потоку початкових TCP-запитів (SYN) та впливу цього потоку на становую підсистему цільового стеку TCP. На рівні характеристик такі можливості охоплюють: (1) формування великої кількості короткоживучих SYN-пакетів з контрольованими параметрами заголовків (порт, розмір, прапорці), (2) масштабування джерел (горизонтальне розгортання великих пулів відправників або проксі-ланцюгів), (3) здатність маніпулювати або

підробляти поле джерела (що ускладнює трасування і змушує сервер створювати стан для «фейкових» сесій) та (4) механізми координації й телеметрії для керування хвилями і оцінки ефекту в реальному часі. Ці характеристики разом визначають, чи буде навантаження більш rps-орієнтованим (великі кількості SYN/с), чи байт-орієнтованим, та яку саме частину ресурсів (таблиці станів, CPU, пам'ять, черги) воно виснажить.

З погляду внутрішньої структури інструментарію, корисно виділити логічні підсистеми: генерація та формування TCP-запитів (функціонал синтезу заголовків і темпів відправлення); інфраструктура розподілених джерел і каналів C2 (для координованих хвиль); механізми маскуванню (маршрутизація через проксі/балансувальники, спуфінг джерела); та модулі зворотного зв'язку (збір телеметрії, адаптація патернів у відповідь на контршляхи). Для аналітиків це означає, що оцінка загрози повинна включати не просто обсяг SYN-трафіку, а й ступінь його «керованості» й адаптивності — чим більше інструментарій підтримує автоматичне перемикавання темпів/цілей, тим складнішим стає виявлення за статичними правилами.

Ключові телеметричні індикатори та метрики, які відображають активність такого інструментарію, включають: темп SYN-пакетів (SYN/s), співвідношення $\text{SYN} \rightarrow \text{SYN-ACK} \rightarrow \text{ACK}$ (наприклад, збільшення частки невідповідених SYN), кількість напіввідкритих (half-open) з'єднань у черзі (SYN backlog usage), rps і bps на граничних інтерфейсах, рівень повторних retransmission та duplicate ACK, частота RST/FIN-пакетів, і збільшення помилок на рівні sockets/NAT/conntrack. Також важлива «анатомія» джерел: розподіл по ASN/географії, ентропія source-port, і наявність великої кількості різних (іноді підроблених) IP-адрес, що породжують SYN. У поєднанні ці показники дозволяють відрізнити легітимні піки (наприклад, flash crowd) від шкідливих хвиль, особливо коли аналіз корелюється з логами балансувальників і додатків.

Для форензики цінність мають: пакетні дампи (pcap) із часовими мітками з кількох точок (edge, балансувальник, кінцевий сервер), записи NetFlow/IPFIX (flow duration, bytes per flow), метрики ядра ОС (схема використання сокетів, помилки accept(), заповнення backlog), логи балансувальників/проксі, а також системні журнали додатків (невдале встановлення сесій, таймаути). Аналіз часових кореляцій між появою SYN-хвиль і збільшенням використання ресурсів дозволяє визначити «момент впливу» і оцінити, які компоненти інфраструктури опинилися в дефіциті (CPU, memory, connection table).

Захист і пом'якшення повинні розглядатися у багатьох вимірах (defence-in-depth):

- Системні/стекові заходи — механізми, що зменшують витрати на напіввідкриті зв'язки (напр., механізми, які дозволяють серверу уникнути створення повноцінного стану на першій SYN, — загальна категорія технік відома в літературі як «SYN cookies» і подібні підходи). Також до цієї групи відносять налаштування TCP-стеку (розмір backlog, таймаути), але практичне тонке налаштування має виконуватись з урахуванням робочого навантаження та тестування в контролі.

- Мережеві/інфраструктурні заходи — фільтрація та rate-limiting на периметрі, застосування SYN-proxies або TCP-terminating балансувальників, будь-які рішення для розподілу навантаження (anycast, CDN) і співпраця з провайдерами (RTBH/Flowspec, напрямок трафіку на scrubber-центри).

- Моніторинг і дедукція — поєднання порогових тригерів (SYN/s vs baseline) з поведінковим аналізом і кореляцією метрик (NetFlow, rpsar, лог балансувальників), щоб уникнути великої кількості хибних спрацьовувань. ML-моделі можуть допомогти в розпізнаванні нетипових патернів, але повинні доповнюватися правилами та контекстними даними.

- Операційні процедури — попередньо підготовлені playbook-и для SOC/NetOps (швидка координація з провайдером, переключення на scrubber, тимчасова зміна політик балансування), drills для відпрацювання реакції, та політики збереження телеметрії для форензики.

Важливим аспектом оцінки «інструментарію» є також характеристики, що ускладнюють захист: широке використання спуфінгу джерел (що знецінює просте блокування по IP), горизонтальне масштабування джерел і адаптивні хвилі (переключення цілей/темтів у відповідь на контрзаходи). Через це локальні одноточкові заходи часто виявляються недостатніми без кооперації на рівні провайдера або без використання розподілених архітектур.

Нарешті, етична та процедурна складова: у дослідницьких роботах та тестах усі моделювання SYN-нагрузок повинні виконуватись тільки в контрольованому середовищі або після отримання письмової згоди власників інфраструктури; результати слід викладати як оцінку вразливостей і рекомендації із захисту, а не як інструкції щодо атак. Для наукового розділу рекомендовано завершувати опис переліком ключових метрик для моніторингу (SYN/s, SYN→SYN-ACK ratio, half-open connection count, SYN backlog utilization, retransmission rate, source ASN dispersion) та практичними рекомендаціями для SOC/NetOps: налаштувати збір NetFlow/IPFIX, зберігати rpsar з критичних точок, мати погоджені процедури координації з провайдерами і регулярно відпрацьовувати playbook-и реагування.

Механізми запобігання та виявлення SYN-flood

Протидія SYN-flood має базуватись на багаторівневому підході, що включає мережеву гігієну, жорстку конфігурацію стеку TCP, інструменти миттєвої фільтрації і архітектурну стійкість.

На мережевому/провайдерському рівні обмеження спуфінгу (ingress/egress filtering, BCP38), uRPF та співпраця з transit-провайдерами зменшують можливість використання підроблених джерел і полегшують трасування. Архітектурні заходи включають розподілену інфраструктуру (Anycast, CDN, георозподілені балансувальники) для розсіювання SYN-навантаження, а також передбачення можливості перенаправлення трафіку на scrubber-центри чи використання managed DDoS-сервісів. На рівні TCP-стеку й серверної конфігурації рекомендується застосування SYN-cookies (замість резервування ресурсу при отриманні SYN), тонке налаштування backlog-параметрів, скорочення таймаутів TCB для незавершених з'єднань, захист від SYN-retry-flood (політики повторних спроб) та відмови від надмірної кількості одночасних напіввідкритих сесій; де можливо, варто використовувати SYN-proxy/connection-proxy (що завершують рукопотискання на периферії) або TCP-proxy, які фільтрують хибні ініціації перед передачею на бекенд. Також корисними є апаратні механізми rate-limiting на рівні граничних маршрутизаторів і L4-фільтрація, token-bucket policing та adaptive rate-limit-policies.

Механізми виявлення поєднують швидкі метрики й поведінковий аналіз: моніторинг SYN/ACK-ratio, співвідношення кількості SYN до встановлених сесій, різке зростання числа напіввідкритих сесій, аномалії в NetFlow/IPFIX (підвищена кількість нових 4-tuple без відповіді) і зміни RTT/latency на критичних інтерфейсах. SIEM-кореляція, порогові алерти для bps/pps і спеціалізовані ML-моделі для часових рядів дозволяють відрізнити flash-crowd від шкідливої кампанії; пакетні сенсори (SPAN/TAP) і IDS/IPS з правилами виявлення аномалій TCP-handshake дають додаткові сигнали. Важливим індикатором є також невідповідність між SYN-потокм і фактичними новими логінами/запитами на прикладному рівні — різкий розрив між мережевою активністю і бізнес-транзакціями вказує на аномалію.

Операційні та процедурні заходи включають попереднє планування (playbooks), узгоджені канали комунікації з провайдерами і mitigator-сервісами, регулярні drills для NetOps/SOC, і механізми швидкого переключення (blackholing/RTBH, reroute на scrubber). Під час інциденту застосовуються комбіновані дії: активація SYN-cookies і SYN-proxy, динамічне rate-limiting, застосування фільтрів на AS-рівні у координації з провайдерами і переміщення трафіку через очищувальні центри;

обов'язковими є збереження flow-даних і рсар для форензики. Після подолання атаки проводиться post-mortem з аналізом MTBD/MTTR, пікових показників (peak bps/pps, peak half-open connections), оцінкою бізнес-втрат і корекцією архітектури (додавання Anycast, переналаштування стеку, оновлення фільтрів).

Підсумовуючи: захист транспортного рівня — це поєднання принципів: мінімізації атакованої площі (закрити непотрібні UDP-сервіси, сувора політика firewall), зміцнення стеку TCP (SYN-cookies, таймаути, TCP-параметри), шифрування та захист сесій (TLS/HTTPS, VPN) і активного моніторингу (NetFlow/NDR/SIEM). Ефективна оборона повинна бути багаторівневою — краєві фільтри й провайдерська підтримка для масштабних DDoS-векторів, внутрішня сегментація та контроль ресурсів для локальних атак, і автоматизовані SOC-процедури для швидкого виявлення й реагування. Такий підхід мінімізує ризик успішного захоплення сесій або виведення сервісів з ладу й дозволяє зберегти доступність і цілісність критичних мережеских сервісів. В умовах еволюції загроз рекомендовано рухатися до принципів zero-trust і SASE (Secure Access Service Edge), запроваджувати мікросегментацію та контроль на рівні додатків, а також використовувати аналітику на основі ML/AI для раннього виявлення аберантної поведінки трафіку. Постійне оновлення інвентарю мережеских сервісів, регулярне патчування і закриття небажаних UDP-портів, а також тісна координація з провайдерами і CERT/CSIRT — це те, що відрізняє підготовлену організацію від вразливої.

3.4. АТАКИ НА ПРИКЛАДНОМУ РІВНІ: SQL INJECTION, XSS, CSRF, COMMAND INJECTION

Атаки на прикладному рівні спрямовані на логіку й обробку даних у веб-додатках і сервісах: вони використовують недоліки валідації, некоректне кодування, небезпечні допуски привілеїв і непередбачені шляхи виконання команд. Хоч ці техніки технічно різні — SQL-ін'єкція, XSS, CSRF, командна ін'єкція — їхня спільна риса в тому, що джерело проблеми зазвичай лежить у довірі до неконтрольованого вводу або в неправильному розмежуванні обов'язків між компонентами системи. У практиці захисту важливо поєднувати превентивні заходи на етапі розробки (secure coding, SAST/DAST, threat modeling), інструменти середовища (WAF, RASP, CSP) і операційні механізми (логування, моніторинг, інцидент-респонс), бо лише комплексний підхід зменшує вірогідність успішної експлуатації вразливостей. Сучасні архітектури (microservices, serverless, GraphQL, SPA) додають нові точки входу і вимагають додаткової уваги до валідації API-

контактів, до безпеки third-party-пакетів (dependency scanning, SBOM) і до захисту DevOps-ланцюгів постачання — уразливість у залежності може відкрити шлях до масштабної компрометації. Окрім того, набирає значення захист даних і приватності у відповідності до регуляторів (GDPR, локальні закони), тому моделі загроз мають включати ризики витоку персональних даних і вимоги до обробки логів. Наявність чіткої класифікації ризиків підсилює можливість пріоритизації виправлень: не всі вразливості рівноцінні — концентрація на критичних функціях і даних дає кращий захисний ефект за обмежених ресурсів.

SQL-ін'єкція (SQL injection) - впровадження SQL-коду

SQL-ін'єкція — атака, що виникає коли небезпечні або неперевірені дані інтерпретуються системою управління базами даних (СУБД) як частина SQL-запиту; атакувальник маніпулює структурою запиту, щоб витягти дані, обійти аутентифікацію або змусити СУБД виконати додаткові команди. Типові варіанти — in-band (union-, error-based), blind (boolean-, time-based) та out-of-band ексфільтрація — і кожен із них дає різні можливості й залишає різні сліди. Наслідки варіюються від витоку конфіденційної інформації й повної компрометації БД до зміни бізнес-логіки й отримання віддаленого доступу через розширення СУБД (наприклад, завантаження файлів або виконання shell-команд). Сучасні загрози також включають NoSQL-ін'єкції і ін'єкції в ORM-запити чи GraphQL-резолвери — тобто захист має охоплювати як класичні SQL-патерни, так і специфіку нових стеків.

Наслідки варіюються від витоку конфіденційної інформації та повного компрометації бази даних до зміни бізнес-логіки й отримання дистанційного доступу через функції СУБД (наприклад, завантаження файлів або виконання shell-команд через розширення). Ефективні контрзаходи — обов'язкове використання параметризованих запитів (prepared statements), ORM з безпечними API, відмова від динамічної конкатенації SQL, сувора валідація/вчищення вводу, обмеження привілеїв облікового запису БД (least privilege), моніторинг підозрілих запитів і налаштування WAF-прав на виявлення характерних шаблонів ін'єкцій. Сучасні загрози також включають NoSQL-ін'єкції та ін'єкції в ORM-запити або GraphQL-резолвери — тому захист має охоплювати не лише класичні SQL-патерни, а й специфічні механізми валідації для нових стеків. Для критичних БД корисно застосовувати Database Activity Monitoring (DAM), аудит транзакцій і шифрування полів з конфіденційними даними, а у випадку виявлення підозрілої активності — швидке відключення облікового запису та ротацію ключів доступу.

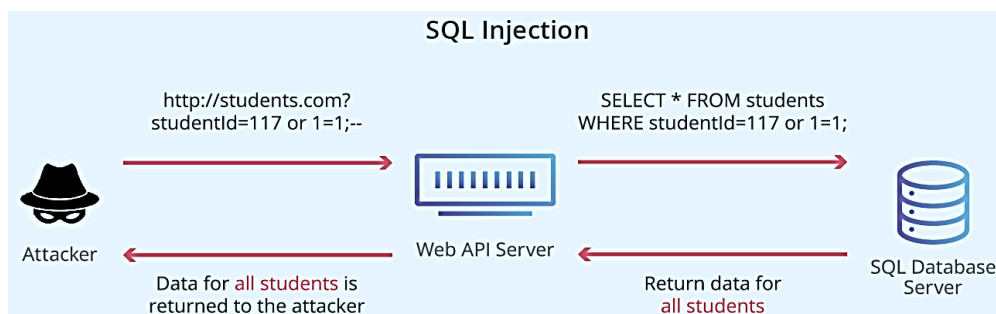


Рис. 3.13. Схема SQL Injection-атаки на веб-додаток
Джерело: [1-9, 11-26, 32-48, 60-65]

Етапи реалізації атаки

Процес побудови атаки на рівні ін'єкційних векторів має вигляд послідовного, але гнучкого циклу розвідки, ідентифікації вразливостей, верифікації гіпотез, експлуатації і подальшого аналізу результатів, причому кожен етап містить як технічні, так і когнітивні компоненти.

На початковому етапі здійснюється розвідка — збір відкритої інформації про ціль: які інтерфейси доступні зовні, які типи форм введення і запитів обробляються, як система реагує на помилки та які конструкції відповіді можуть надати підказки про внутрішню логіку. Ця фаза не обмежується лише автоматизованими сканерами; важливу роль відіграє аналіз поведінки застосунку, вивчення шаблонів помилок, кешування, форматів відповідей і навіть тимчасових характеристик реакції, оскільки всі ці сигнали допомагають моделювати ймовірні шляхи, якими зовнішній ввід може потрапити в механізм формування запиту до СУБД.

На етапі ідентифікації вхідних точок атаки досліджуються конкретні поля форм, параметри URL, заголовки та API-параметри, які можуть впливати на структуру запитів; особлива увага приділяється всім місцям, де застосунок оперує даними користувача без попередньої перевірки або трансформації. Далі йде верифікація гіпотез про вразливість — контрольовані, етичні спроби викликати зміну поведінки системи, щоб визначити, чи дозволяє механізм формування запиту зовнішньому вводу змінити логіку.

Важливо підкреслити, що всі експерименти повинні відбуватися лише в ізольованому тестовому середовищі або за письмовою згодою власника системи; будь-які дії проти продуктивних систем без дозволу є протиправними. Якщо гіпотеза підтверджується, настає концептуальна фаза оцінки можливостей експлуатації: визначення, які дані можуть бути витягнуті, які привілеї може надати компрометація, чи існують шляхи ескалації прав або взаємодії з іншими компонентами інфраструктури.

Паралельно з цим проводиться оцінка слідів і ризиків — які дії залишать помітні логи, які механізми моніторингу можуть показати атаку, і які перешкоди (ліміти, WAF, таймаути) обмежують масштаб та швидкість ексфільтрації. Якщо атака іде в фазу пост-експлуатації, дослідник концептуально оцінює наслідки компрометації: можливість зміни бізнес-логіки, витоку персональних даних або отримання віддаленого виконання команд через розширення СУБД; ця інформація використовується для підготовки відповідних рекомендацій і плану реагування.

Завершальною частиною процесу є документування знайдених вразливостей, формалізація відтворюваного сценарію у безпечному вигляді та передача висновків власникам системи з чітким планом усунення — при цьому головний принцип відповідального розкриття має бути дотриманий невідкладно та етично.

Інструментарій прикладного рівня, релевантний до сценаріїв SQL-ін'єкції

Інструментарій прикладного рівня, релевантний до сценаріїв SQL-ін'єкції, слід розглядати як сукупність функціональних можливостей, поведінкових патернів і архітектурних умов у стеку «клієнт → веб-застосунок → шар бізнес-логіки → СУБД», які полегшують або ускладнюють можливість неприпустимого впливу на запити до бази даних. До ключових характеристик цього інструментарію належать: спосіб прийому й обробки вхідних даних (параметрів форм, заголовків, API-параметрів), наявність/відсутність централізованої валідації і нормалізації, модель формування SQL-запитів (рядкова конкатенація vs. параметризовані запити), рівень логування запитів і помилок, а також політики привілеїв облікових записів бази даних. Інструментарій також включає компоненти проміжної інфраструктури — проксі, кеші, ORM-шари, шаблонізатори та мікросервіси — які формують контекст і часто змінюють «тон» запитів (наприклад, динамічні запити в ORM чи конструктори SQL у сервісах).

З погляду аналітика загроз важливо описувати не конкретні програми, а класи функцій: автоматичні сканери та fuzz-модуля, які генерують величезні обсяги варіативних параметрів; проксі-перехоплювачі, що дозволяють модифікувати трафік у реальному часі; лог-агрегаційні системи та АРМ-агенти, які дають видимість у виконання запитів; і модулі, що формують динамічні SQL-шаблони. Кожен із цих класів має свою «діагностичну відбитковість»: сканери створюють часті повтори з малою варіативністю, проксі-перехоплювачі змінюють заголовки й параметри в потоках, а ORM-шари генерують складні запити із шаблонами, які мають типові підписки в логах.

Індикатори компрометації і ранні сигнали, пов'язані з активністю інструментарію прикладного рівня, мають як мережево-логу-орієнтований, так і прикладний характер. Серед них — різке зростання кількості помилок бази даних (SQL-error rates) або HTTP-відповідей 500/422, особливо за рахунок однакових URI з різними довгими/нестандартними значеннями параметрів; аномально довгі або структурно незвичні значення параметрів (висока ентропія/довжина поля); нетипова послідовність запитів до певного endpoint-у; поява повільних або довготривалих SQL-запитів; збільшення частки failed/rollback транзакцій; і розбіжності між логами застосунку та очікуваною бізнес-логікою (наприклад, операції змінюють більшу кількість рядків, ніж передбачалося). Також корисними є поведінкові сигнали: множинні різні значення параметра від одного IP/UA за короткий проміжок часу, або збільшення унікальних значень параметра із малою глибиною взаємодії (ознака автоматизованого сканування).

Джерела телеметрії і форензичні артефакти, які потрібно агрегувати для достовірного аналізу, включають: веб-серверні логи (з параметрами запитів, User-Agent, IP, referer), логи проміжних проксі/балансувальників, application-level логи з трасуванням транзакцій і параметрів, журнали СУБД (slow query log, general query log, audit log), записи APM/trace (розподілені траси), та SIEM-агреговані події. Для післяінцидентної реконструкції особливо цінні повні рядки виконаних SQL-запитів (якщо їх безпечно зберігають) з часовими мітками, контекстні журнали аутентифікації/авторизації, бекапи транзакцій та дампи пам'яті додатка (у законному й етичному порядку). Важливо планувати retention-політику так, щоб критичні артефакти були доступні під час розслідування.

Захисні характеристики й підходи впливають безпосередньо з природи інструментарію: найефективніші заходи мінімізують можливість формування небезпечних запитів на джерелі і підвищують видимість аномалій на трасі. Технічно-архітектурні рекомендації включають: централізовану валідацію і нормалізацію вхідних даних (принцип allow-list), використання параметризованих запитів і підготовлених виразів (prepared statements) у всіх місцях формування запитів, впровадження ORM-парадигми з усвідомленою політикою безпечного формування запитів або явним биндингом параметрів, відмова від динамічної рядкової конкатенації SQL, обмеження можливостей виконання кількох запитів в одному виклику, застосування принципу least privilege для облікових записів DB (мінімальні права на виконання лише необхідних операцій), і включення механізмів аудитування на рівні СУБД. Також до оборонних шарів належать WAF із позитивними правилами (allow-list) і контекстними сигнатурами, runtime application self-protection (RASP) для виявлення аномалій на рівні

виконання, та системи rate-limiting/blacklisting для підозрілих pattern-запитів.

Окрема група заходів — це процесні й організаційні практики: статичний і динамічний аналіз коду (SAST/DAST) у CI/CD-пайплайні, регулярні code-review і threat modeling, security-by-design підхід при розробці API (чіткі схеми, типізація, обмеження довжини полів), навчання розробників (secure coding), і програмне забезпечення оновлень та патчів для всіх залежностей. Тестування на вразливості має виконуватися лише у контрольованих умовах або за письмовою згодою власника ресурсу; результати тестів мають документуватися з рекомендаціями щодо виправлення і перевірки.

Для виявлення на ранніх стадіях доцільно налаштувати спеціальні тригери в SIEM/APM: підвищення рівня помилок SQL (порівняно з базовою лінією), різке збільшення кількості унікальних значень параметра за короткий період, spike у довжині/ентропії параметрів, а також поява незвичних комбінацій endpoint+метод (наприклад, POST до endpoint-у, який типово отримує короткі числові id). Рекомендовані метрики для моніторингу: error rate (DB errors/min), ratio failed/total queries, median і p95 виконання запитів, number of unique parameter values per minute, average parameter length, частка довготривалих транзакцій, а також частота змін рядків у критичних таблицях. Кореляція цих метрик із NetFlow, балансувальниками та логами аутентифікації підвищує точність виявлення.

Нарешті, етичний і юридичний контекст є невід'ємною частиною опису: будь-яке сканування, тестування або симуляція SQL-ін'єкцій повинні бути санкціоновані, проводитись у лабораторіях або на стендах і супроводжуватись політикою зберігання доказів та обробки персональних даних. В аналітичному матеріалі доцільно завершити розділ коротким переліком «індикаторів → джерела → рекомендовані контрзаходи», а також вказівкою на політики відповіді (playbook) для SOC, які визначають кроки збору телеметрії, координацію з командами розробки й процедури повідомлення відповідних стейкхолдерів.

Механізми запобігання та виявлення ін'єкційних атак

Протидія таким загрозам потребує багатоварового підходу, що поєднує технічні, процесні і організаційні заходи захисту. На рівні розробки архітектури та коду головним пріоритетом є виключення можливості інтерпретації неконтрольованого вводу як частини запиту: це досягається через систематичне застосування параметризованих запитів і механізмів прив'язки параметрів, відмову від динамічної конкатенації SQL-рядків і використання ORM з безпечними API там, де це доречно; при цьому варто уникати «raw»-запитів без суворої валідації.

Санітизація вводу повинна базуватися на підході whitelisting — перевірці типу, довжини, формату і допустимих значень, а не на спробах відфільтрувати всі потенційні шкідливі послідовності. Роль принципу найменших привілеїв критична: облікові записи додатків мають мати лише ті права, що необхідні для їх роботи, а адміністративні доступи і операції з виконанням команд СУБД повинні бути суворо розділені й контролюватися. Корисним підходом є також застосування транзакційних або процедурних інтерфейсів (stored procedures) за умови, що вони розроблені без логічних недоліків; проте слід пам'ятати, що самі процедури можуть містити уразливості, якщо в них допускається небезпечна обробка вводу.

Інфраструктурний і моніторинговий шари додають наступний рівень захисту: розгортання Web Application Firewall (WAF) і правил для виявлення характерних шаблонів ін'єкцій може суттєво ускладнити масовані або автоматизовані спроби, хоча WAF не замінює безпечний код. Системи Database Activity Monitoring (DAM) і логування запитів забезпечують видимість нетипової активності — великих експортів, частих помилок, аномалій у шаблонах виконання або звертань у незвичний час. SIEM-інструменти дозволяють корелювати події з різних джерел і створювати алерти по порогах і поведінковим аномаліям, що дає змогу реагувати на інциденти на ранніх стадіях. Техніки обмеження ексфільтрації, такі як rate limiting, throttle на API, ліміти на обсяги результатів і застосування CAPTCHA для підозрілих сценаріїв, знижують ефективність автоматизованих атак. Для даних підвищеної чутливості рекомендується шифрування на рівні полів (field-level encryption) або токенизація — навіть у випадку компрометації таблиць дані будуть недоступні без ключів.

Процеси розробки теж мають бути захищені: інтеграція безпеки в SDLC, регулярні code review, застосування статичного аналізу (SAST) для виявлення антипатернів у роботі з БД, динамічне тестування (DAST) у контрольованому середовищі і авторизовані пентести допомагають знайти проблеми до релізу. Важлива частина захисту — навчання розробників і інженерів операцій: розуміння загроз, принципів безпечної роботи з БД і типових помилок значно зменшує ймовірність появи вразливостей. Нарешті, на рівні операцій і реагування необхідно мати відпрацьовані плани інцидент-менеджменту: чіткі процедури ізоляції, збереження доказів, ротації облікових даних і ключів, резервного відновлення та комунікації з зацікавленими сторонами і регуляторами. Поєднання цих технічних і організаційних заходів у рамках підходу defense-in-depth і постійної оцінки загроз дозволяє суттєво знизити ризики ін'єкційних атак та мінімізувати потенційні наслідки у випадку їх реалізації.

Cross-Site Scripting (XSS) — міжсайтовий XSS-скриптинг

Cross-Site Scripting (XSS) — це клас вразливостей, при яких зловмисний скрипт доставляється до браузера жертви через веб-сторінку і виконується в контексті довіри до цього сайту. Існують три основні підвиди: збережений (stored) — шкідливий скрипт зберігається на сервері і потім віддається іншим користувачам; відбитий (reflected) — скрипт повертається у відповіді на запит (наприклад, через параметр URL); DOM-based — виникає виключно на клієнтському боці через небезпечну обробку DOM API.

XSS порушує межі довіри між клієнтом і сервером та підриває модель same-origin, що може призводити до викрадення сесій, ініціації небажаних запитів від імені користувача, ексфільтрації даних і стійкої персистенції шкідливого коду в інтерфейсі. Ступінь потенційної шкоди визначається контекстом виконання (наприклад, HTML-текст, атрибут, JavaScript-контекст, CSS або URL) і набором привілеїв, доступних у середовищі браузера (cookie, localStorage, доступ до DOM-API, розширення). Надійна детекція й запобігання вимагають контекстно-чутливої ескейпінга (output encoding), політик Content Security Policy (CSP) та суворої валідації вводу в поєднанні з інтеграцією безпеки в SDLC і автоматизованими SAST/DAST-сканерами. З позицій формальної оцінки ризику XSS може розглядатися як клас клієнтських ін'єкційних вразливостей, модельованих через множину точок входу, векторів поширення та контекстів виконання, що дозволяє формалізувати метрики ризику й оцінити ефективність контрзаходів.

Етапи реалізації атаки

Етапи реалізації атаки Cross-Site Scripting (XSS) мають чітку структурну послідовність, яка поєднує як технічні, так і когнітивні дії й за своєю суттю є життєвим циклом експлуатації вразливості клієнтського боку.

На першому етапі проводиться розвідка: аналіз цільового веб-застосунку для виявлення інтерфейсів введення та місць, де дані користувача відображаються в HTML, JavaScript або інших контекстах клієнтського виконання (формові поля, параметри URL, заголовки, коментарі тощо). Ця фаза включає вивчення типової поведінки сторінок при помилках, характеру відгуку сервера й динаміки DOM, а також виявлення меж між серверним і клієнтським рендерингом — оскільки різні класи XSS (reflected, stored, DOM-based) мають різні точки входу й механізми поводження.

Другий етап — ідентифікація вхідних точок і контекстів відображення — зосереджується на встановленні того, чи потрапляє неперевірений ввід у

HTML-текст, атрибути, JavaScript-контекст, CSS або URL; саме контекст визначає допустимі техніки атаки і потенційний вплив (сесійне викрадення, підміна контенту, фішингові інтерфейси, ескалація привілеїв у межах браузера).

Третій етап — верифікація гіпотез — передбачає контрольовані спроби змінити вивід і спостереження за поведінкою клієнта та сервера; метою є підтвердження наявності відображення вводу у вразливий контекст або виявлення динамічних змін DOM, які можуть призвести до DOM-орієнтованої ін'єкції. Далі йде концептуальна фаза оцінки наслідків: визначення можливостей для крадіжки сесійних маркерів, підміни UI для фішингу, ініціації небажаних запитів від імені користувача або перехоплення локального сховища; у разі виявлення збережуваного XSS — аналіз шляхів персистенції та потенційних масштабів поширення шкідливого контенту. Пост-експлуатаційні дії з академічної точки зору включають оцінку слідів і ризиків, визначення шляхів ескалації і lateral movement у межах веб-інфраструктури, а також підготовку звіту з відтворюваним описом вразливості й рекомендаціями щодо її усунення з дотриманням принципів відповідального розкриття; нагальне застереження — будь-які практичні тестування мають проводитися лише в авторизованому середовищі з письмовою згодою власника ресурсу.

Наслідки XSS включають викрадення сесійних токенів, фішингові сторінки під ім'ям легітимного ресурсу, несанкціоновані дії від імені користувача і розповсюдження шкідливого коду. Захист фундаментально будується на принципі «не довіряй вводу, ескейпуй при виводі»: контекстне кодування вихідних даних (HTML-encode, attribute-encode, JS-escape, URL-encode), використання шаблонних рушіїв з автоматичним екрануванням, політика Content Security Policy (CSP) з дефініцією джерел і nonce-токенами, маркування cookies як HttpOnly/Secure, застосування суворих заголовків (X-Content-Type-Options, X-Frame-Options) та мінімізація використання небезпечних функцій (innerHTML, eval). На додаток, практики як регулярне застосування sanitizer-бібліотек (наприклад, DOMPurify), налаштування CSP reporting для збору інцидентів XSS у продакшні і активне застосування SRI (Subresource Integrity) для підключених скриптів підвищують рівень безпеки. Важливо також тестувати клієнтську логіку: автоматизовані IAST/DAST-скани, fuzzing полів і ручні рев'ю фронтенд-шаблонів виявляють небезпечні місця, де фреймворк-екранування може бути обійдено.

Stored cross-site scripting

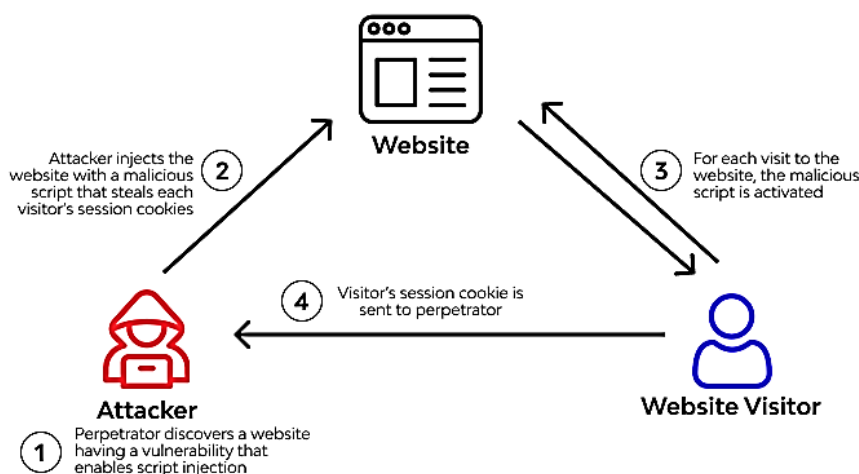


Рис. 3.14. Схема XSS-атаки

Джерело: [1-9, 11-26, 32-48, 60-65]

Інструментарій прикладного рівня, релевантний до ризиків XSS

Інструментарій прикладного рівня, релевантний до ризиків XSS, слід розглядати як сукупність компонентів стека «клієнт → веб-інтерфейс → бізнес-логіка → постійне сховище → рендеринг», а також пов'язаних інструментів розробки і моніторингу, які формують середовище виконання й можуть як пом'якшувати, так і посилювати ймовірність появи XSS-інцидентів. До таких компонентів належать: механізми прийому і нормалізації вхідних даних (форми, API-ендпоїнти, файли завантаження); шаблонізатори і бібліотеки рендерингу (із або без автоматичного ескейпінгу); ORM/конструктори запитів, які можуть зберігати «сировинні» рядкові поля; клієнтський код (JS/SPA-фреймворки), що маніпулює DOM; а також проміжна інфраструктура — проксі, кеші, CDN і WAF — які впливають на видимість і поведінку запитів. Важливо також виділити зовнішні інтеграції (вбудовувані віджети, third-party scripts, markdown-рендерери), які змінюють модель довіри під час рендерингу контенту.

Ключові характеристики такого інструментарію: контекст рендерингу (вставка в HTML-текст, атрибут, URL, JavaScript context, CSS), політика ескейпінгу (чи виконується вона автоматично шаблонізатором і чи залежить від контексту), наявність механізмів фільтрації/санітизації на вході, політика привілеїв зберігання (хто має право писати в поля, які потім рендеряться для інших користувачів), логування операцій запису/читання і підтримка механізмів аудиту. Саме ці характеристики визначають, чи потенційно небезпечний контент може зберігатися та потім виконуватись у браузері користувача.

Індикатори та телеметрія, які допомагають виявляти або підозрювати XSS-активність, мають прикладний і поведінковий характер. Серед таких індикаторів: різке зростання помилок рендерингу або HTTP-помилки (наприклад, 500) при однотипних URI з різними параметрами; аномальна ентропія або довжина вхідних полів; поява у базі даних значень, що містять HTML-теги або скриптоподібні фрагменти в тих полях, які раніше зберігалися тільки як простий текст; сплески повідомлень CSP-report (якщо CSP увімкнено і збирає репорти); часті запити до сторінок із великою кількістю унікальних значень параметрів від одного IP/UA (ознака сканування); та нетипові дії користувачів після відображення контенту (наприклад, сесійні дії, які не відповідають UI-подіям в логах). Для повнішої картини корелюють веб-логи (access logs), application logs (включно з контентом параметрів), DB-аудит (stored values), WAF-логи і CSP-репорти.

Форензика XSS-інцидентів зазвичай спирається на збережені артефакти: дампи HTTP-запитів і відповідей (з часовими мітками), записи бази даних із відповідними полями, логи застосунку з трасуванням транзакцій, WAF/IDS-логі та CSP-report-payloads, журнали CDN і проксі, а також записи APM/trace, які показують послідовність виконання коду у момент відображення контенту. Особливо корисними є: (1) збережені оригінальні вхідні параметри (raw request), (2) збережені варіанти після нормалізації, (3) хронологія збережень у БД (хто і коли записав контент), та (4) співставлення часу появи запису і часу перших клієнтських подій, що можуть свідчити про експлуатацію. Політика збереження цих артефактів повинна бути налаштована заздалегідь, щоб не втратити сліди під час реагування.

Заходи захисту впливають безпосередньо з аналізу інструментарію: найефективніші підходи мінімізують шанс виконання шкідливого контенту на боці клієнта й посилюють видимість аномалій. До сукупних методів належать: контекстно-чутливе кодування/ескейпінг вихідних даних (encoding для HTML text, HTML attribute, JavaScript string, URL context і т. ін.); білістова (allow-list) валідація і нормалізація вхідних даних; заборона використання рядкової конкатенації для формування HTML/JS у серверних шаблонах; застосування шаблонізаторів з автозахистом (auto-escaping) і явним дозволом на вбудування безпечного HTML лише через перевірені механізми; використання Content-Security-Policy (CSP) з режимом звітування (report-only) і поступовим посиленням політик; HttpOnly та Secure атрибути cookies, SameSite-налаштування і обмеження scope куків; обмеження привілеїв запису (які сторінки/ролі можуть зберігати HTML у полях); WAF і RASP як додаткові шари виявлення/пом'якшення; а також

підтримка і обробка CSP-репортів у SIEM для швидкого виявлення потенційних спроб виконання небезпечного контенту.

Окремий набір протидій стосується процесів розробки та тестування: інтеграція SAST (аналіз коду на етапі коміту) для виявлення небезпечних конкатенацій/незахищених своїх викликів, DAST/IAST у середовищі тестування для виявлення відображених і збережених XSS-патернів за контрольованими сценаріями, рев'ю шаблонів і бібліотек рендерингу, контроль third-party скриптів і бібліотек (SCA — software composition analysis), а також політики secure-by-default у фреймворках. У CI/CD має бути заборона на просування у виробництво коду з відомими XSS-вразливостями та автоматизовані гейти (tests/quality gates).

Для SOC/NetOps і SIEM корисно визначити набір детекторів і метрик: частота помилок 4xx/5xx по певним endpoint-ам; число унікальних значень параметра/URI за одиницю часу; кількість і зміст CSP violation reports; зростання частки записів у БД, що містять HTML/теги; spike у довжині/ентропії параметрів; підвищений rate запитів із однаковими payload-шаблонами (ознака сканування). Тригери мають поєднувати порогові правила з кореляцією подій із різних джерел (waf + db + app + csp), щоб зменшити хибні спрацювання.

Операційні процедури при підозрі на XSS повинні включати: ізоляцію і збереження щодо сумнівних записів (take a forensic snapshot — DB row, request logs, app logs), збір CSP-reports і pcap/HTTP dumps із часовими мітками, кореляцію з подіями аутентифікації і діями користувачів, застосування тимчасових обмежувальних політик (rendering disabled / sanitization enforced / WAF rule temporary tighten) до вразливого функціоналу, та повідомлення розробників і запуск поправки у тестове середовище. Важливі також регулярні drills і post-mortem з метриками MTTD/MTTR, impact on users і рекомендаціями щодо змін у CI/CD і runtime.

Механізми запобігання та виявлення ін'єкційних XSS-атак

Механізми запобігання та виявлення ін'єкційних XSS-загроз формується як багаторівнева система технічних, процесних і організаційних контрзаходів, орієнтована на зменшення площі атаки та раннє виявлення аномальної активності. На рівні коду й архітектури ключовими є принципи контекстної ескейпінга (екранізації) та розмежування вводу й представлення: усі дані зовнішнього походження повинні екрануватися відповідно до контексту виводу (HTML-текст, атрибути, JavaScript-строки, URL, CSS), застосовуватися автозахист фреймворків і шаблонізаторів, що забезпечують безпечне відображення за замовчуванням, і уникатися безпечної практики вставки неперевірених даних у DOM через innerHTML або аналогічні методи; замість цього рекомендується використовувати

textContent, safe APIs і декларативні механізми, які не інтерпретують введення як код. Валідація вводу має базуватися на whitelisting допустимих форматів і строго типізуванні, а не на спробах вилючити потенційні шкідливі конструкції; додатково до валідації на стороні сервера слід застосовувати клієнтські перевірки як дружній UX, але не як єдиний бар'єр.

Інфраструктурні контрзаходи включають застосування політик Content Security Policy (CSP) для обмеження джерел скриптів, стилів і інших ресурсів, використання nonces або hash-підписів для дозволених inline-елементів та суворої відмови від unsafe-inline і unsafe-eval у продуктивних профілях; CSP також слугує інструментом для виявлення (report-uri/report-to) підозрілих порушень політики. Захист сесій і конфіденційних маркерів реалізується через HTTP-Only та Secure куки, встановлення атрибута SameSite, мінімізацію або заборону зберігання секретів у локальному сховищі та впровадження механізмів обмеження життєвого циклу сесій. На рівні мережі й периметра корисними є Web Application Firewall (WAF) з правилами для виявлення характерних ознак XSS-атак, проте WAF розглядається як додатковий шар, а не як заміна безпечного коду.

Механізми виявлення і моніторингу поєднують техніки логування, кореляції подій і аналітики поведінки: налаштування журналювання подій серверу та клієнтських помилок, збір звітів CSP, кореляція повідомлень у SIEM для виявлення аномалій (раптові помилки парсингу, хвилі запитів із подібними параметрами від різних джерел, зростання повідомлень CSP), а також періодичний аналіз контенту, що зберігається, для виявлення небезпечних вставок у довготривалій пам'яті (stored XSS). Процеси розробки й тестування доповнюють технічні засоби: інтеграція безпеки у SDLC, статичні та динамічні сканери (SAST/DAST) для виявлення потенційних місць ризику, регулярні авторизовані пентести з увагою на DOM-аналіз і тестування сценаріїв, а також code review з чек-лістами щодо безпечного рендерингу. Не менш важливі організаційні заходи — навчання розробників і тестувальників, формалізація політик щодо використання сторонніх бібліотек та компонентів (щоб уникнути внесення уразливого коду), встановлення процедур реагування на інциденти з визначеними ролями, планом containment, збором доказів і повідомленням зацікавлених сторін. В сукупності ці контрзаходи, реалізовані за підходом defence-in-depth і підкріплені регулярним аудитом і оновленням політик, суттєво зменшують імовірність реалізації XSS-атак та мінімізують їх потенційні наслідки.

CSRF (Cross-Site Request Forgery) — підробка міжсайтових запитів»

CSRF (Cross-Site Request Forgery) — атака, що використовує те, що браузер автоматично докладає авторизаційні cookie або інші креденціали до запитів: атакуючий підштовхує автентифікований користувачеві браузер виконати небажану дію на сайті жертви (наприклад, переказ коштів, зміну пароля). CSRF відрізняється від XSS тим, що тут атакувальник не виконує код у контексті сторінки жертви, а змушує жертву самостійно надіслати запит; тому суть захисту — переконатися, що запит походить від легітимного інтерфейсу.

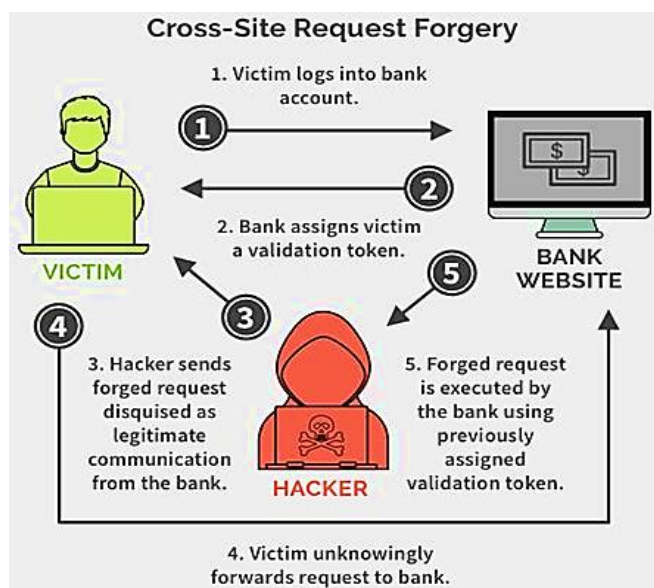


Рис. 3.15. Схема атаки Cross-Site Request Forgery (CSRF)
Джерело: [1-9, 11-26, 32-48, 60-65]

Етапи реалізації атаки

Етапи реалізації атаки (життєвий цикл CSRF). Cross-Site Request Forgery (CSRF) — це клас атак, під час яких атакувальник змушує автентифікованого у цільовому сервісі користувача ініціювати небажану HTTP-операцію від імені цього користувача, використовуючи наявні сесійні облікові дані та контекст довіри між браузером і сайтом.

Життєвий цикл такої атаки починається з розвідки: аналізу поведінки цільового сайту для виявлення state-changing endpoint'ів (операції зміни стану, наприклад POST/PUT/DELETE або GET із побічними ефектами), механізмів аутентифікації (cookie, заголовки авторизації) та очікуваних параметрів запитів.

Далі відбувається фаза підготовки — створення «носія» атаки (шкідлива сторінка, зображення, форма, скрипт на сторонньому ресурсі або фішинговий лист), що містить запит із потрібними параметрами; важливо, що цей запит повинен бути таким, щоб браузер жертви при відвідуванні або

рендері зовнішнього ресурсу автоматично відправив його до цільового домену разом із релевантними куками/авторизацією. Тестування й верифікація відбуваються в контрольованому середовищі: перевіряється чи призводить відправка подібного запиту до очікуваної дії на сервері (зміни профілю, переказ коштів, зміна пароля тощо); при цьому атаку слід імітувати лише в авторизованих умовах. У випадку успіху настає фаза експлуатації — здійснення атаки проти реальних користувачів (масова доставка шкідливих посилань або розміщення шкідливого контенту на ресурсах, ймовірно пов'язаних з жертвами).

Паралельно формується канал моніторингу й логування: налаштовуються збірні точки (веб-логи, WAF, CSP-репорти, SIEM) для оперативного виявлення аномалій у state-changing запитах і кореляції подій. Перед будь-якою масовою доставкою шкідливого вмісту проводиться юридична перевірка й отримується письмовий дозвіл власників інфраструктури — усі тестові дії мають виконуватися лише в рамках погодженого пентесту або лабораторних стендів. У разі початку експлуатації команда реагування заздалегідь активує playbook: тимчасово обмежує або ізолює уразливий функціонал, посилює валідацію вхідних даних і застосовує тимчасові контрзаходи на рівні WAF та інфраструктури, а також координується з провайдерами для мінімізації шкоди. Після інциденту проводиться ретроспектива з аналізом MTTD/MTTR, оновленням CI/CD-гейтів, політик безпеки й процедур, а також передача висновків відповідним командам для впровадження стійких контрзаходів.

Завершальною є фаза пост-експлуатації та аналізу — оцінка масштабу ураження, збір артефактів (журнали, відповіді сервера), визначення точок слабкості в логіці авторизації і підготовка звіту з рекомендаціями щодо ліквідації уразливостей і запобігання повторенню інциденту. Важливо підкреслити, що, на відміну від ін'єкційних атак, CSRF експлуатує модель довіри між браузером і сервером (автоматична відправка cookie), отже найуспішнішими атаки будуть тоді, коли сайт використовує cookie-базовані сесії без додаткових захисних механізмів або допускає зміну стану через методи, що легко тригераються зовнішніми сторінками.

Інструментарій прикладного рівня, релевантний до ризиків CSRF

Інструментарій прикладного рівня, релевантний до ризиків CSRF, найкраще описувати не як набір конкретних програм, а як класи функціональних можливостей і архітектурних характеристик стеку «клієнт → веб-інтерфейс → бізнес-логіка → бекап/сервіси», які визначають, наскільки вразливою є здатність застосунку виконувати небажані зміни стану від імені автентифікованого користувача. У центрі уваги — механізми аутентифікації (cookie-based sessions, persistent cookies, token-based auth),

модель управління сесіями й токенами, способи прийому й обробки state-changing запитів (POST/PUT/DELETE), політики CORS, наявність/відсутність захисних маркерів (CSRF-токенів), а також клієнтський код (форми, AJAX-запити, SPA-фреймворки), що формує і відправляє запити. До інструментального контексту також належать проміжні компоненти (reverse proxy, CDN, WAF), які змінюють або приховують заголовки (Referer/Origin), і third-party інтеграції (вбудовувані віджети, сторонні скрипти), що можуть бути джерелом непередбачуваної взаємодії між доменами.

Ключові характеристики такого «інструментарію» визначаються поведінкою застосунку та клієнта: чи використовує застосунок cookie-автентифікацію для state-changing API (залежність, що підвищує ризик CSRF), чи застосунок очікує специфічні заголовки або токени у запиті, чи існує захист на рівні форми (hidden CSRF token) і чи прив'язаний цей токен до сесії/користувача. Важливим є також контекст відправлення запиту (HTML-форма, IMG/IFRAME, XHR/fetch) — різні механізми мають різні обмеження для крос-доменної відправки й тому по-різному впливають на можливість підробки запитів.

Індикатори та телеметрія, що вказують на потенційну активність або проблеми з CSRF-захистом, мають як мережево-логовий, так і прикладний характер. Сигналами ризику є: значна кількість state-changing запитів (POST/PUT/DELETE) без валідації CSRF-токена або зі зламаними/невалідними токенами; часті помилки валідації токенів у короткий проміжок часу; невідповідність заголовків Origin/Referer очікуваним доменам при запитах, що змінюють стан; повторювані запити з однаковими payload від різних реферерів; та розбіжності між логами UI-подій (кліки) і фактично виконаними операціями (наприклад, запис у базі без відповідної UI-дії). Для раннього виявлення корисно поєднувати WAF-логування (відхилені запити), application-level логи (результати валідації токенів), web-server access-логи (заголовки, IP, UA, referer/origin) і SIEM-кореляцію.

Форензичні артефакти, корисні для розслідування інциденту, включають: сирі HTTP-логи із заголовками (особливо Cookie, Referer, Origin, User-Agent), збережені payloads POST-форм, журнали аудиту дій у застосунку (хто і коли ініціював зміну), записи сесій (session id), WAF/CSP-reports, та записи CDN/proxy. Важливо забезпечити збереження цих артефактів з точними часовими мітками і можливістю кореляції між різними джерелами (сервер, балансувальник, CDN, клієнтські логи). Реконбінація даних дозволяє відрізнити легітимну дію користувача від підробленого запиту, що використовував вже установлену сесію.

Захисні заходи впливають на пряму з розуміння інструментарію: першочергове значення має усунення умов, що дозволяють браузеру автоматично відправляти аутентифікаційні атрибути у крос-доменної взаємодії. Практичні захисні патерни (в академічному викладі) включають: застосування Synchronizer Token Pattern (унікальні, криптографічно стійкі CSRF-токени, прив'язані до сесії або запиту, що перевіряються на сервері), Double-Submit Cookie (дублювання токена в cookie і в параметрі/заголовку та порівняння), налаштування атрибута SameSite для cookie (Lax/Strict) для обмеження автоматичного відправлення куків при крос-доменної навігації, валідацію заголовків Origin/Referer для state-changing запитів, вимогу використання Authorization-header (Bearer token) для API замість cookie-автентифікації, а також застосування повторної аутентифікації або додаткової користувацької валідації для критичних операцій. Для SPA-архітектур важливо реалізувати захист при роботі з XHR/fetch (перевірка custom headers або CSRF-токенів в заголовках), ретельну конфігурацію CORS та відмову від прийому state-changing запитів через прості HTML-форми із зовнішніх доменів.

Операційні практики для зниження ризику включають: інтеграцію перевірок CSRF у тестову автоматизацію (DAST/IAST у CI/CD), регулярні code-review та threat-modeling під час розробки API, моніторинг метрик в SIEM (rate of CSRF validation failures, number of state-changing requests without Origin/Referer, spikes in POST requests from external referers), і готові playbook-и для реагування (ізоляція підозрілих сесій, тимчасове посилення політик, повідомлення відповідальних розробників). Важливо балансувати чутливість тригерів та керувати хибними спрацьовуваннями, оскільки надмірні блокування можуть порушити легітимну функціональність.

Механізми запобігання та виявлення CSRF-атак

Механізми запобігання та виявлення CSRF (комплексні контрзаходи). Захист від CSRF вимагає поєднання архітектурних рішень, практик розробки, конфігурації протоколів і моніторингу. На рівні проєктування та коду ключовим є принцип відмови від використання станозмінних операцій через прості GET-запити; всі операції, що змінюють стан, повинні застосовувати безпечні методи і перевіряти семантику запиту. Основні превентивні техніки включають застосування синхронізованих анти-CSRF токенів (унікальних, непердаваних автоматично браузером значень, які сервер очікує у тілі або заголовку запиту) і патерну double-submit cookie у випадках, коли це доречно; ці підходи гарантують, що запит підписаний або містить секрет, доступний лише через контекст сторінки, а не через зовнішній ресурс. Атрибути cookie також надають потужні механізми: конфігурування SameSite (Lax або Strict, за вимогами UX) може істотно

знизити ризик CSRF шляхом заборони автоматичної відправки сесійних cookie під час навігації з зовнішніх сайтів, а атрибути HttpOnly і Secure мінімізують ризики витоку і перехоплення. Перевірки джерела запиту (Origin або Referer) можуть слугувати додатковим бар'єром, однак їх не слід розглядати як єдину лінію захисту через можливі проблеми сумісності та випадки, коли ці заголовки відсутні або спотворені; краще поєднувати їх з криптографічними анти-CSRF токенами. Для API та SPA корисні стратегії: використання токенів доступу в заголовках (наприклад, Authorization Bearer), відмова від збереження критичних маркерів у cookie при відсутності належного захисту SameSite, а також вимога спеціального заголовку X-Requested-With або схожого маркера, який брався б із сторінки, а не надсилався автоматично браузером при сторонніх навантаженнях. Крім технічних заходів, важливо проектувати UX так, щоб критичні операції (переклади, зміна прав доступу) вимагали додаткової явної взаємодії користувача (наприклад, підтвердження паролем, двофакторна автентифікація або повторне введення критичних даних), що ускладнює автоматизовану експлуатацію.

Моніторинг і виявлення доповнюють превенцію: систему логування слід конфігурувати для детального реєстрування state-changing запитів (час, джерело, заголовки Origin/Referer, відсутність/наявність анти-CSRF токена), а SIEM-кореляція може ідентифікувати аномалії — наприклад, великі хвилі запитів із однаковою payload-структурою або звернення від унікальних user-agent/реферер-комбінацій. WAF-правила можуть виявляти типові патерни атак і блокувати спроби ініціювати несанкціоновані дії, але їх слід використовувати як додатковий шар. Для API-інтерфейсів рекомендовано виконувати поведінковий аналіз (anomaly detection) — виявлення незвичних шаблонів операцій для користувача або клієнта (наприклад, кількість транзакцій за короткий інтервал) і авто-тримування (throttling) чи додаткове верифікування. Процедури реагування мають включати механізми швидкої ізоляції (відклик токенів, ротація сесійних ключів), форензичний збір доказів та повідомлення за встановленими політиками.

Організаційні та процесні заходи завершують оборонну модель: інтеграція захисту від CSRF в SDLC (чек-лісти для code review, автоматизовані тести у CI), регулярні DAST-сканування й авторизовані пентести, навчання розробників щодо правильного використання cookie-політик і анти-CSRF патернів, а також документовані політики щодо мінімізації дій, які можуть викликатися автоматично. У сукупності ці технічні, моніторингові і організаційні контрзаходи реалізують концепцію

defence-in-depth, суттєво зменшуючи імовірність успішної CSRF-експлуатації та обмежуючи її потенційні наслідки.

Найпоширеніші механізми захисту: використання анти-CSRF токенів (унікальних, не вгадуваних значень, прив'язаних до сесії і перевірюваних для кожного state-changing запиту), техніка double-submit cookie, перевірка заголовків Origin/Referer для критичних запитів, запровадження SameSite атрибута для cookie (Strict або Lax) та вимога додаткової взаємодії (re-auth, CAPTCHA, 2FA) для чутливих дій. Сучасні SPA і мобільні клієнти вносять свої нюанси: використання токенів у localStorage робить додаток вразливим до XSS, тому схеми збереження токенів і одночасний захист від XSS мають розглядатися разом. Також важливо коректно налаштувати CORS: при відкритих або надмірно широких CORS-політиках захист від CSRF може стати неефективним.

Командна ін'єкція (command injection)

Командна ін'єкція (command injection) — це небезпека, коли дані від користувача потрапляють у виклики системних утиліт або shell-команд без належної фільтрації чи розділення аргументів, і через це атакувальник здатен виконати довільні команди на хості. Це трапляється при побудові рядків для функцій типу system(), popen(), exec() або при використанні shell-інтерпретаторів для обробки вхідних параметрів; варіанти включають пряму ін'єкцію, використання керованих файлів конфігурації або обхід обмежень через спецсимволи й опції.

Етапи реалізації атаки

Етапи реалізації атаки. Командна ін'єкція — це клас вразливостей, при яких неконтрольований ввід користувача потрапляє в контекст виконання команд операційної системи або інтерпретатора на стороні сервера і призводить до виконання небажаних операцій поза межами проектної логіки. Життєвий цикл реалізації такої атаки починається з розвідки: аналізу застосунку для виявлення точок, де зовнішні дані можуть передаватися в інтерфейси виконання команд (наприклад, системні виклики, виклики шелу, інтерфейси для завантаження файлів, обробки зображень або інтеграції з утилітами ОС). На цій стадії збирають інформацію про привілеї процесу, середовище виконання, наявність оболонок, шаблонів формування запитів і зворотних повідомлень, які можуть допомогти змодельовати можливі шляхи впливу — усе це з метою коректного оцінювання атакувальної поверхні без спроб фактичного виконання шкідливих команд у продуктивному середовищі.

Наступний етап — ідентифікація вхідних точок і контекстів інтерпретації: з'ясування, чи проходить ввід через функції або бібліотеки, що делегують виконання ОС (виклики shell/exec), чи формується рядок

команд динамічно, чи використовуються тимчасові файли, пайпи або інші межпроцесні механізми. Верифікація гіпотез здійснюється виключно в ізольованих лабораторних умовах або в рамках авторизованого тестування і має на меті визначити, чи можливе ін'єкційне впровадження керуючих символів, параметрів або контекстів, що змінюють поведінку інтерпретатора; на цьому етапі також оцінюється можливість ескалації привілеїв і бокового переміщення в інфраструктурі. У разі підтвердження вразливості йде концептуальна фаза оцінки наслідків — визначення обсягу даних, які можуть бути витягнуті або модифіковані, доступних файлових системних ресурсів, можливостей для довготривалого доступу (backdoor), а також потенційного впливу на цілісність і доступність сервісу. Завершальним кроком цього життєвого циклу є документування результатів, підготовка рекомендацій щодо ліквідації виявлених недоліків і передача цих результатів власнику системи у рамках відповідального розкриття; особливо важливо фіксувати артефакти та логи для форензики і планувати подальші заходи з моніторингу та відновлення.

Наслідки — від видалення даних і викрадення конфіденційної інформації до повного захоплення сервера й ескалації привілеїв. Протидія — суворі валідація та біл-лістинг допустимих значень, відмова від виклику shell коли можливо (заміна на API, бібліотеки або виклики з передачею аргументів як масиву без інтерпретації), безпечні функції виклику процесів (execve з аргументами), ескейпінг у разі неминучого використання shell, розгортання у least-privileged контексті (не запускати веб-процеси від root), контейнеризація, обмеження системних викликів (seccomp), та моніторинг/логування підозрілих викликів. До цього додаються сучасні ризики — небезпечна десеріалізація, ін'єкції у CI/CD скрипти або уявлення шаблонів (template injection), які теж можуть призвести до виконання команд — отже, рев'ю і тестування pipeline-скриптів та перевірка прав виконання середовищ є обов'язковими.

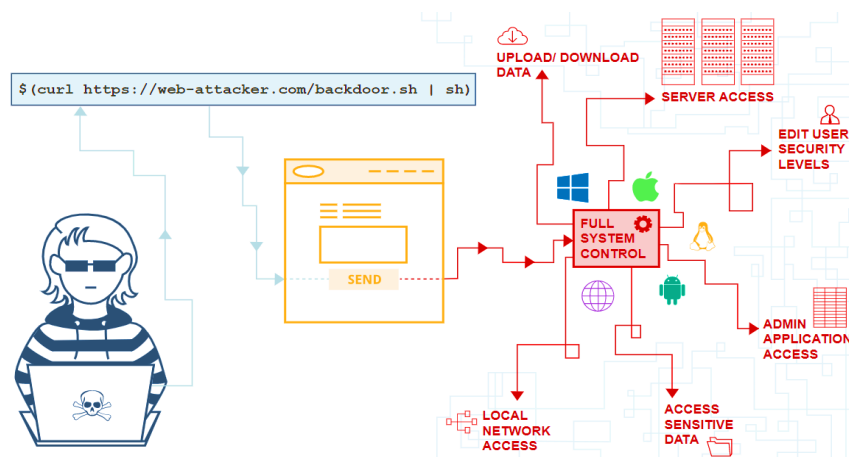


Рис. 3.16. Схема командної ін'єкції (Command Injection)

Джерело: [1-9, 11-26, 32-48, 60-65]

Інструментарій прикладного рівня, що робить застосунок вразливим до командної ін'єкції

Інструментарій прикладного рівня, що робить застосунок вразливим до командної ін'єкції або опосередковано пов'язаний з цією загрозою, — це не перелік окремих «хакерських» утиліт, а сукупність функціональних можливостей, архітектурних рішень і патернів коду, які дозволяють зовнішньому введенню бути інтерпретованим як команда в оболонці або як виклик системних утиліт на сервері. До таких характеристик належать: використання API або фреймворків, які допускають виконання системних команд або інтерпретацію рядків як команд (наприклад, виклики виконання команд на боці сервера через обгортки в мовах/бібліотеках); формування команд шляхом рядкової конкатенації з необроблених вхідних даних; наявність серверного коду з підвищеними привілеями (root/адмін) або дефіцитом ізоляції між процесами; використання тимчасових файлів і відкритих записів у файловій системі без належної перевірки; інтеграція з зовнішніми службами, які можуть обробляти і повторно інтерпретувати дані (shell-перетворення, шаблонізатори зі «виконуваною» вставкою); та недостатній рівень логування й аудитування дій, що ускладнює оперативне виявлення і реконструкцію інциденту.

З практично-аналітичної точки зору важливо виділити класові елементи «інструментарію», які заслуговують на увагу аналітика загроз і захисника: вхідні точки — форми, API-параметри, файли завантаження, заголовки, cookie-поля, зовнішні інтеграції; шари інтерпретації — шаблонізатори, конструктори викликів, ORM/конструктори запитів, оболонки команд у коді; контрольно-керівні модулі — логування, валідація, механізми escaping, політики привілеїв; операційні контейнери — середовища виконання (контейнери, вірт. машини), де ізоляція або політики SELinux/AppArmor можуть бути відсутніми або неправильно налаштованими; інструменти розробки/CI — SAST/DAST, які або є в пайплайні, або відсутні. Саме поєднання цих елементів створює «поверхню атаки» для можливих спроб підміни вхідних даних на командні виклики.

Індикатори компрометації та аномальні сигнали, які породжує активність, пов'язана з командною ін'єкцією, мають багатовимірний характер — мережево-логовий, прикладний і хостовий. Серед таких індикаторів: поява в логах веб-серверу або застосунку довгих або структурно нетипових значень параметрів; spike у кількості помилок 500/4xx після специфічних запитів; аномальні виклики системних бібліотек

з боку веб-додатку; несподівані створення тимчасових файлів або зміни в критичних конфігураційних файлах; поява нетипових дочірніх процесів, яких не було в нормальній роботі сервісу; різкі стрибки CPU/IO, network egress (вихідних з'єднань) з хоста у моменти після специфічних запитів; та розбіжності між подіями UI (натискання кнопок) і фактичними змінами в системі. Важливо корелювати ці сигнали: поодинокі аномалії можуть бути легітимними, але їх сукупність (наприклад, незвичний параметр → новий процес → outbound connection) підвищує ймовірність інциденту.

Джерела телеметрії й форензичні артефакти, які треба зберігати та аналізувати при підозрі на командну ін'єкцію: повні HTTP/HTTPS-логи з таймстемпами та повними рядками параметрів (raw request), application logs із трасуванням виконання (stack traces, context), системні журнали (syslog, auditd), процесні зрізи (ps, procfs-метрики), записи створених процесів і їхніх аргументів (process accounting), файли створених тимчасових ресурсів, логи контейнерів/оркестратора (Docker, Kubernetes events), NetFlow/IPFIX для кореляції з мережевою активністю, а також дампи пам'яті та знімки файлової системи (з дотриманням правових процедур) для відтворення подій. Наявність централізованого збирання й довготривалої ретеншн-політики значно підвищує шанси успішної реконструкції інциденту.

Захисні підходи впливають безпосередньо з характеристик інструментарію. Архітектурні й кодові рекомендації (в академічному, нефункціональному вигляді) включають: сувору білістову (allow-list) валідацію й нормалізацію вхідних даних за типами й довжиною; відмову від рядкової конкатенації для формування команд і використання безпечних, високорівневих API, які не інтерпретують рядки як оболонкові команди; принцип найменших привілеїв для процесів, що виконують запити (ізолювані облікові записи, drop capabilities, use of sandboxing); уникнення зберігання секретів у середовищах зниженої ізоляції; застосування механізмів обмеження створення процесів і network egress; обмеження прав файлової системи (read-only для ділянок, де це можливо); використання контейнеризації з правильними політиками безпеки (seccomp, AppArmor/SELinux профілі); та забезпечення ретельного логування всіх викликів до системних інтерфейсів. У контексті БД/файлових операцій — мінімізувати функції, які дозволяють виконувати командні операції з контенту (наприклад, avoid executing arbitrary interpretable content).

Інструменти забезпечення якості й верифікації безпеки мають постійно інтегруватись у SDLC: SAST для виявлення небезпечних конструкцій у коді (наприклад, місць конкатенації вхідних даних у виклики виконання), DAST/IAST у тестових середовищах для виявлення аномальної поведінки при форсованих input-сценаріях, code reviews з фокусом на

безпеку, та політики захисту залежностей (SCA). CI/CD-гейти повинні блокувати промоцію коду з виявленими критичними проблемами і мати автоматизовані тести на input validation. WAF та RASP можуть служити додатковим шаром виявлення/пом'якшення, але не замінюють виправлення кореневих причин у коді.

Операційні та SIEM-правила (без конкретних payload-ів) доцільно формулювати як кореляційні детектори: поєднувати аномалії в HTTP-лозі (параметри нетипової довжини/ентропії) з появою невластивих дочірніх процесів, зростанням outbound connections або незвичними файловими операціями; тригери на часті помилки 500 після однакових або схожих запитів; оповіщення про виконання процесів під обліковим записом сервісу, що зазвичай не створює дочірніх процесів; та аномалії в поведінці мережі (несподіваний трафік до зовнішніх адрес). Важливо визначити рівні чутливості й механізми автоматичного збирання додаткових артефактів (рсар, процесні знімки) при активації високого рівня підозри.

Форензика й реакція на інцидент: при підозрі на командну ін'єкцію одразу «законсервувати» релевантні артефакти (raw request, app logs, процесні зрізи, файлові снапшоти), ізолювати процес/сервіс у безпечному режимі або відключити його від мережі, зберегти деталі середовища виконання (версії бібліотек, конфігурації), та провести ретельну кореляцію часових міток. Після відновлення необхідно провести root-cause analysis (як уразливий вхід потрапив до точки інтерпретації), оновити політики доступу, закласти рев'ю й тести, що запобігатимуть повторенню, і провести post-mortem із метриками (MTTD, MTTR, бізнес-вплив).

Механізми запобігання та виявлення командних ін'єкцій

Механізми запобігання та виявлення командних ін'єкцій формують багаторівневу стратегію, що поєднує принципи безпечного програмування, конфігураційну жорсткість і активний моніторинг виконання. На рівні проєктування і коду центральним положенням є відмова від викликів інтерпретатора ОС шляхом використання безпечних API: якщо потрібно виконати функціонал (наприклад, робота з файлами, зображеннями, мережевими операціями), слід застосовувати бібліотеки, що надають відповідні можливості без делегування у shell; при цьому уникнення динамічної конкатенації рядків для формування команд є фундаментальною вимогою. Валідація вводу має базуватися на принципі whitelist — обмеженні допустимих форматів, типів і значень, а також на строгій типізації і нормалізації даних; крім того, слід застосовувати контекстно-залежну санітизацію для випадків, коли частина вхідних даних мусить бути передана у зовнішні інтерфейси. Практикою зниження ризику є реалізація механізмів ізоляції виконання: запуск зовнішніх процесів у обмежених

контейнерах або sandbox'ах з мінімальними правами, обмеження набору дозволених бінарників і шляхи до них, використання chroot/jail- або контейнерних політик, обмеження можливостей мережевого та файлового доступу для процесів. Принцип найменших привілеїв має застосовуватися послідовно — сервіси й процеси повинні працювати під мінімально необхідними обліковими записами, а доступ до критичних ресурсів (коди, ключі, конфігураційні файли) повинен бути ретельно контрольований.

Інструментальні заходи включають статичний аналіз коду (SAST) для виявлення патернів потенційно небезпечних викликів виконання команд, динамічне тестування (DAST) у тестових середовищах, контроль виконання в рантаймі (RASP) і використання інтрузійних систем та систем виявлення аномалій на рівні хосту та додатків. Логування і моніторинг виконання процесів, аргументів команд і аномалій у роботі процесів мають бути налаштовані так, щоб корелювання з SIEM давало змогу виявляти незвичні виклики або нестандартні параметри запуску; також корисними є системи аналізу поведінки процесів (behavioral monitoring), які можуть сигналізувати про підозрілі дії (раптове створення з'єднань, незвична активність файлової системи, виникнення дочірніх процесів). На рівні інфраструктури доцільно застосовувати обмеження на мережеві виклики і egress-фільтрування, щоб унеможливити ексфільтрацію даних у разі компрометації, а також централізоване управління дозволеними пакетами програмного забезпечення (application allow-listing).

Процедурні та організаційні контрзаходи охоплюють інтеграцію безпеки в життєвий цикл розробки (SDLC): код-рев'ю з фокусом на ризики виконання команд, регулярні автоматизовані сканування залежностей і оновлення компонентів, авторизовані пентести та перевірки налаштувань середовища. Навчання розробників і операторів щодо ризиків делегування виконання ОС, шаблонів безпечної роботи з зовнішніми утилітами та правил управління привілеями знижує ймовірність помилкових рішень у проектуванні. У разі інциденту план реагування має передбачати швидку ізоляцію уражених компонентів, збір артефактів для форензики (журнали, дампи процесів), ротацію сесійних ключів і облікових даних, а також відновлення з перевірених резервних копій. Комплексне застосування описаних заходів у рамках defence-in-depth дозволяє зменшити ймовірність експлуатації командних ін'єкцій до прийняттого рівня ризику та скоротити потенційні наслідки у разі їх реалізації.

Для виявлення атак прикладного рівня корисні як статичні, так і динамічні підходи: SAST виявляє потенційно небезпечні конструкції в коді на ранній стадії, DAST/pen-testing симулює атаки на працюючу систему, а RASP і WAF дають runtime-захист і миттєву блокаду підозрілих запитів.

Незалежно від технології, корисні практики — централізоване логування запитів і помилок, моніторинг аномалій (наприклад, незвична кількість помилок 500, підозрілі рядки у параметрах), SSO та MFA для підвищення бар'єрів доступу, а також регулярне тестування (DAST, IAST) і програми bug-bounty, що дозволяють виявляти проблеми до їхнього експлуатації. Додатково ефективні механізми — структуроване логування (JSON), трасування запитів (distributed tracing), інтеграція RASP-сигналів у SIEM/UEBA для кореляції, та використання «saпaгу»-токенів і honeypot-ендпойнтів для раннього виявлення зловмисної активності. Після інциденту пріоритети — ізоляція уражених компонентів, ротація сесій і паролів, forensic-збір (логи, дампи), аналіз вектору атаки і швидке виправлення (patching, оновлення бізнес-логіки), а також комунікація з постраждалими та, за потреби, регуляторами.

Підсумовуючи, захист від атак на прикладному рівні — це не лише набір технічних патчів, а життєвий цикл безпеки: threat modeling при проєктуванні, secure-by-default бібліотеки і фреймворки, автоматизовані перевірки під час CI/CD, навчання розробників, обмеження привілеїв на всіх рівнях і постійний моніторинг. Сучасний стек вимагає додаткових практик: сканування залежностей (SCA), генерація і підтримка SBOM, IaC-сканування (Terraform/CloudFormation), секрет-менеджмент і обмеження доступу в runtime середовищі. Лише поєднання правильних архітектурних рішень (розділення прав доступу та мінімізація функцій у привілейованих компонентах), безпечного кодування (параметризовані запити, контекстне екранування) та оперативної готовності реагувати на інциденти забезпечує довготривалу стійкість додатків проти SQLi, XSS, CSRF і командних ін'єкцій. Розвиток «security as code» (сканери в CI, gated deploys, автоматизовані поліметрики TTD/TTR) робить захист масштабованим і дозволяє знижувати ризики без значного уповільнення реліз-циклів.

3.5. ПРОТОКОЛИ ТА МЕХАНІЗМИ ЗАХИСТУ МЕРЕЖ: VPN, IPSEC, MPLS, ZERO TRUST NETWORK ACCESS

Протоколи та механізми захисту мереж відіграють ключову роль у забезпеченні конфіденційності, цілісності та доступності передаваних даних. Вони створюють базис для безпечної взаємодії між користувачами, сервісами та додатками, навіть у розподілених і хмарних інфраструктурах. Сучасні рішення включають багаторівневе шифрування, контроль доступу, моніторинг аномалій і засоби реагування на інциденти в реальному часі. Зі зростанням кількості мобільних і IoT-пристроїв ці протоколи адаптуються, впроваджуючи легші криптографічні алгоритми та підтримку

масштабованих моделей аутентифікації. Важливим трендом є інтеграція традиційних механізмів захисту з концепціями Zero Trust і SASE, що дозволяє забезпечувати безпеку незалежно від місця підключення чи пристрою користувача.

VPN (Virtual Private Network) забезпечує захищений «тунель» через публічну або загальнодоступну мережу, шифруючи трафік між кінцевими точками — користувачем і корпоративним шлюзом або між двома офісами. Існує кілька реалізацій VPN (SSL/TLS-базовані, IPsec-VPN, WireGuard тощо): SSL/VPN зручні для віддалених користувачів через веб-інтерфейс і гнучкі політики доступу, але потребують уважного налаштування TLS і валідації сертифікатів; IPsec забезпечує більш формалізовані, стандартизовані тунелі для site-to-site з детальною конфігурацією алгоритмів шифрування й аутентифікації.

При розгортанні VPN важливо застосувати сильну аутентифікацію (багатофакторну аутентифікацію), оновлювані криптографічні набори, політики розмежування доступу та логування сесій для подальшого аудиту. Сьогодні на практиці дедалі більше впроваджують сучасні й легкі рішення на кшталт WireGuard, які дають кращу продуктивність і простішу конфігурацію при збереженні високого рівня безпеки. Також зростає використання тимчасових (ephemeral) ключів і короткотривалих сертифікатів для зменшення ризику компрометації сесій. Не можна ігнорувати ризики split-tunneling — для деяких сценаріїв його слід забороняти або суворо контролювати політиками. У контексті підготовки до квантової криптографії організації починають аналізувати вплив PQC (post-quantum cryptography) на VPN-інфраструктури та планувати оновлення криптопровайдерів.

VPN як метод захисту мережевих комунікацій

Virtual Private Network (VPN) — це технологія створення захищеного «тунелю» поверх публічних або незахищених мереж, яка забезпечує цілісність і конфіденційність даних шляхом шифрування та автентифікації учасників. VPN дозволяє встановлювати захищені канали як між кінцевим користувачем і корпоративним шлюзом, так і між двома або більше офісами, що розташовані у віддалених сегментах мережевої інфраструктури. Існує кілька основних реалізацій: SSL/TLS-VPN, що зручні для віддаленого доступу через веб-інтерфейси та мобільні клієнти; IPsec-VPN, які забезпечують стандартизовані site-to-site з'єднання з можливістю точного налаштування криптографічних алгоритмів; та сучасні рішення на кшталт WireGuard, які поєднують високу продуктивність із простотою конфігурації.

Залежно від архітектури можна виділити кілька моделей VPN. Remote Access VPN призначені для індивідуальних користувачів і підтримують багатофакторну автентифікацію, політики доступу та контроль сесій. Site-to-Site VPN застосовуються для постійного об'єднання офісів і дата-центрів у єдину логічну мережу з централізованим управлінням. Cloud-VPN інтегруються з хмарними платформами, забезпечуючи захищений доступ до сервісів AWS, Azure чи GCP. Кожна з цих моделей має специфічні вимоги до налаштування шифрування, керування ключами та розмежування доступу.

Методи підвищення стійкості VPN включають використання сучасних криптографічних наборів (AES-256, ChaCha20-Poly1305), короткотривалих сертифікатів і тимчасових (ephemeral) ключів, які знижують ризик компрометації тривалих сесій. Важливо забезпечувати постійний моніторинг стану VPN-інфраструктури: логування підключень, аналіз аномальної поведінки (одночасні логіни з різних геолокацій, нетипові години доступу), інтеграцію з системами SIEM і автоматизованими засобами реагування SOAR. Додатково застосовуються механізми Network Access Control (NAC), які перевіряють стан клієнтського пристрою перед встановленням тунелю.

Водночас використання VPN має низку обмежень і ризиків. Однією з ключових загроз є єдина точка відмови — у випадку компрометації VPN-шлюзу зловмисник може отримати доступ до всієї корпоративної мережі. Значний виклик становить і проблема split-tunneling, коли частина трафіку йде напряму у відкритий Інтернет, минаючи захисні механізми, що потребує чітких політик і контролю. Крім того, VPN накладають додаткове навантаження на інфраструктуру через постійне шифрування і можуть знижувати продуктивність у великих корпоративних мережах.

Сучасний розвиток безпеки мережевих комунікацій виводить VPN на новий рівень. Організації дедалі частіше інтегрують VPN у концепцію Zero Trust Network Access (ZTNA), де перевірка відбувається не лише під час встановлення тунелю, а й на рівні кожного запиту до ресурсу. Розробники і провайдери впроваджують механізми post-quantum cryptography (PQC) для підготовки VPN-протоколів до майбутніх викликів квантових обчислень. Додатково поширюються архітектури Software Defined Perimeter (SDP), які дозволяють гнучкіше керувати доступом і створювати віртуальні сегменти, приховані від сторонніх учасників мережі.

Таким чином, VPN залишаються базовим інструментом захисту інформаційних систем, але їхнє безпечне застосування потребує не лише правильного налаштування шифрування й автентифікації, а й інтеграції з сучасними концепціями кіберзахисту, моніторингу та управління ризиками.

IPsec (Internet Protocol Security) — набір протоколів для забезпечення конфіденційності, цілісності та аутентичності IP-пакетів на мережевому рівні. IPsec використовує IKE (Internet Key Exchange) для встановлення безпечних ключів, а потім ESP/AH для шифрування й перевірки цілісності. Його сильні сторони — можливість прозорого захисту для додатків без змін у їхній конфігурації, стандартизованість і добре протестовані механізми тунелювання та транспортного режимів. Однак IPsec складніший у адмініструванні (політики, SA, профілі IKE), чутливий до проблем NAT і іноді потребує додаткових рішень для масштабування (наприклад, використання concentrator/terminator-пристроїв).

Сучасні впровадження IPsec широко використовують IKEv2 з підтримкою MOBIKE для мобільних клієнтів і AEAD-шифри (наприклад, AES-GCM, ChaCha20-Poly1305) для кращої продуктивності та стійкості. Для подолання проблем NAT застосовують NAT-Traversal (UDP encapsulation) та балансувальники сесій, а у великих середовищах — апаратні криптоускорювачі для зниження навантаження на CPU. Також зростає практика поєднувати IPsec із централізованим управлінням ключами (PKI) та оркестрацією конфігурацій через API/CI-CD. Важливим є застосування стандартів і рекомендацій (RFC) та регулярне оновлення SA-політик у відповідь на нові вразливості.

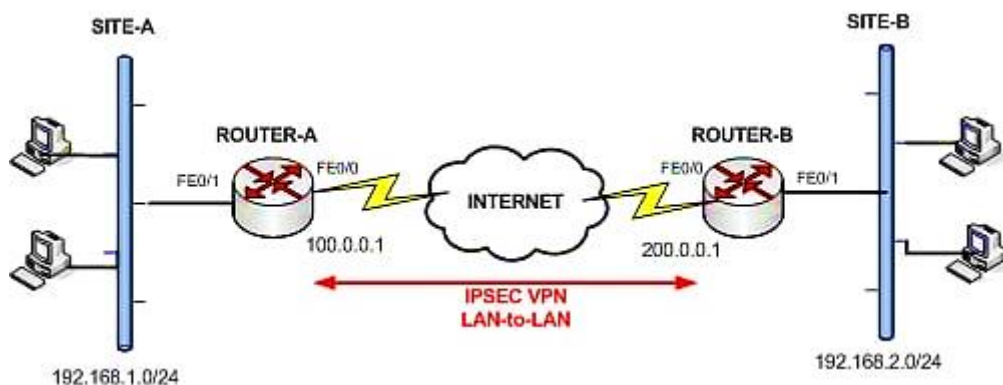


Рис. 3.17. Архітектура тунелю IPsec для з'єднання локальних мереж через Інтернет)

Джерело: [1-9, 11-26, 32-48, 60-65, 72-77]

IPsec як базова технологія захисту на мережевому рівні

IPsec (Internet Protocol Security) — це набір протоколів, розроблений для забезпечення конфіденційності, цілісності та автентичності IP-пакетів безпосередньо на мережевому рівні. На відміну від засобів прикладного рівня, IPsec працює прозоро для додатків і не потребує змін у їхній конфігурації, що робить його універсальним рішенням для побудови

захищених каналів. Центральними компонентами є протокол обміну ключами IKE (Internet Key Exchange), який встановлює безпечні ключі та параметри сеансу, а також протоколи ESP (Encapsulating Security Payload) і АН (Authentication Header), що реалізують шифрування та перевірку цілісності. IPsec може функціонувати у двох режимах: транспортному (захист лише даних у пакеті) та тунельному (захист усього IP-пакета з інкапсуляцією), що забезпечує гнучкість у різних сценаріях.

Архітектурні сценарії застосування IPsec охоплюють як site-to-site VPN для безпечного з'єднання віддалених офісів і дата-центрів, так і remote access VPN для мобільних користувачів, які підключаються до корпоративної мережі через клієнтські програми. У великих мережах IPsec часто комбінують із GRE-тунелями для підтримки мультипротокольного трафіку, а також інтегрують із SD-WAN для динамічної маршрутизації й оптимізації каналів. Завдяки стандартизованості IPsec є сумісним із більшістю мережевих пристроїв, але практичне розгортання потребує ретельного налаштування політик, Security Associations (SA) та профілів IKE.

Механізми безпеки IPsec постійно вдосконалюються. Сучасні впровадження базуються на IKEv2, який підтримує MOBIKE (Mobility and Multihoming Protocol) для стійкої роботи мобільних клієнтів, а також використовують AEAD-алгоритми шифрування (AES-GCM, ChaCha20-Poly1305), що поєднують конфіденційність і цілісність у межах одного процесу та підвищують продуктивність. Для подолання проблем із NAT активно застосовується NAT-Traversal (UDP-інкапсуляція), а у великих середовищах — апаратні криптоускорювачі, які знижують навантаження на центральні процесори.

Важливою практикою є інтеграція IPsec із централізованими системами управління ключами (PKI) та автоматизація конфігурацій через API або CI/CD-підходи. Це дозволяє масштабувати інфраструктуру, швидко оновлювати політики та зменшувати ризик людських помилок. Особливу увагу слід приділяти регулярній ротації ключів, короткоживучим Security Associations та моніторингу IPsec-тунелів у системах SIEM, що допомагає своєчасно виявляти аномалії (підозрілі спроби IKE, нетипові відключення або повторні ініціації).

Разом із сильними сторонами IPsec має і певні обмеження. Його впровадження є складнішим в адмініструванні порівняно з SSL/TLS-базованими VPN: потрібні чіткі політики безпеки, коректна синхронізація SA та управління сертифікатами. Крім того, IPsec чутливий до сумісності різних реалізацій — обладнання від різних вендорів може відрізнятися в

деталіях підтримки RFC. Масштабованість також потребує використання concentrator/terminator-пристроїв, що виступають у ролі вузлів агрегації.

Сучасні тенденції розвитку IPsec пов'язані з інтеграцією в Zero Trust Network Access (ZTNA), коли захищений канал є лише частиною багаторівневої моделі доступу з постійною перевіркою контексту користувача й пристрою. У поєднанні з архітектурою Software Defined Perimeter (SDP) IPsec використовується як базовий транспортний механізм у динамічно керованих середовищах. Зростає інтерес і до впровадження post-quantum cryptography (PQC) у рамках IPsec/IKE для підготовки інфраструктури до квантових загроз.

Отже, IPsec є фундаментальною технологією захисту мережевого рівня, що забезпечує високий рівень конфіденційності та цілісності даних. Його ефективність визначається правильним налаштуванням політик, інтеграцією з сучасними системами управління та моніторингу, а також готовністю до нових викликів — від масштабування до переходу на постквантові алгоритми.

MPLS (Multiprotocol Label Switching) — технологія транспортного рівня, яка маршрутизує трафік на основі ярликів (labels), а не традиційних IP-маршрутів, що дозволяє реалізовувати логічні розділені мережі (VPN на рівні оператора — MPLS-VPN), забезпечувати гарантовану пропускну здатність, пріоритизацію та QoS для критичних сервісів. У порівнянні з публічними VPN, MPLS-VPN дає операторам і підприємствам більш передбачувану продуктивність та розмежування трафіку без необхідності шифрування на рівні провайдера; якщо потрібен захист конфіденційності на каналі через недостатню довіру до провайдера, поверх MPLS можна додати IPsec-тунелі. MPLS корисний для великих мереж із жорсткими вимогами до SLA, VoIP і мультимедіа, але його недолік — залежність від інфраструктури провайдера й вартість у порівнянні з інтернет-тунелями.

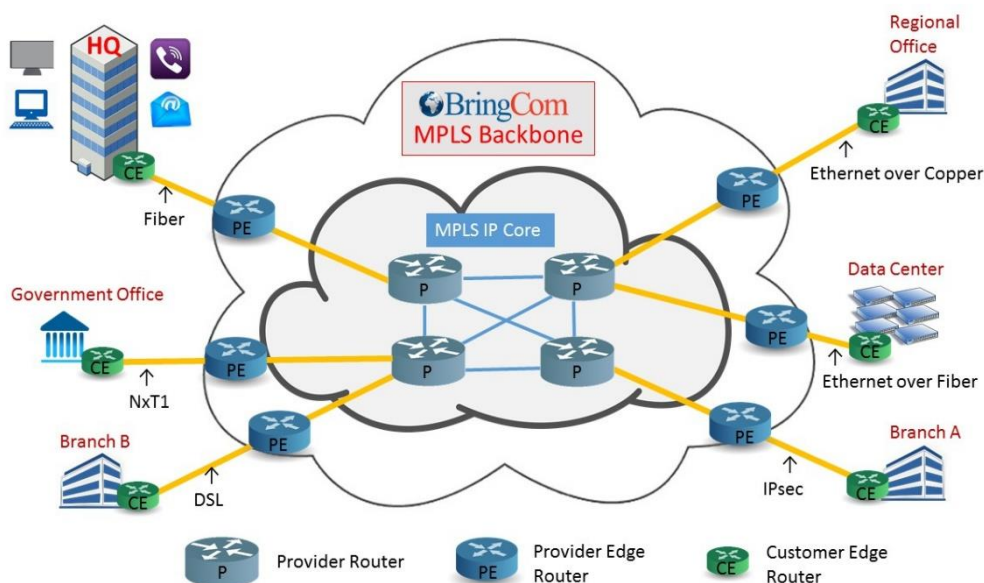


Рис. 3.18. Топологія MPLS із підтримкою різних каналів доступу (Fiber, DSL, Ethernet, IPsec)
Джерело: [1-9, 11-26, 32-48, 60-65, 72-77]

Сучасні оператори розвивають EVPN для мульти-контейнерної віртуалізації L2/L3 і впроваджують Segment Routing (SR-MPLS) для спрощення керування маршрутами й підвищення стійкості. Також дедалі частіше MPLS комбінують із SD-WAN, що дозволяє динамічно балансувати трафік між MPLS і Інтернетом за якістю та вартістю. Для посилення безпеки використовують захист контрольної площини (RTBH, uRPF), шифрування на середині мережі та моніторинг BGP/Label-флешей. Важливим трендом є планування шляху до переходу частини функціоналу в хмарні SASE-сервіси, де MPLS може співіснувати з глобальними розподіленими шлюзами.

MPLS у забезпеченні безпеки корпоративних мереж

MPLS (Multiprotocol Label Switching) — це технологія транспортного рівня, яка маршрутизує трафік на основі ярликів (labels), а не класичних IP-маршрутів. Такий підхід забезпечує ефективну комутацію пакетів і дозволяє реалізовувати логічно розділені мережі з високою масштабованістю. Основними перевагами MPLS є підтримка VPN на рівні оператора (MPLS-VPN), можливість гарантованої пропускну здатності, пріоритизації й управління якістю обслуговування (QoS), що робить цю технологію особливо цінною для критичних сервісів, таких як VoIP, мультимедіа чи транзакційні системи.

MPLS підтримує кілька моделей впровадження. Layer 2 VPN (VPWS, VPLS) дозволяють прозора з'єднувати віддалені сайти на каналному рівні,

формуючи віртуальну Ethernet-мережу поверх операторської інфраструктури. Layer 3 VPN реалізують логічно розділені маршрутизовані домени з використанням механізмів VRF (Virtual Routing and Forwarding), що забезпечує масштабоване розділення клієнтського трафіку та зручність адміністрування. Таким чином, підприємства отримують гнучкий вибір моделі залежно від архітектурних потреб.

З точки зору безпеки MPLS має подвійний характер. З одного боку, технологія забезпечує ізоляцію трафіку між клієнтами та підвищену передбачуваність маршрутизації. З іншого — вона не гарантує криптографічного захисту, а отже, у середовищах із підвищеними вимогами до конфіденційності рекомендується додатково застосовувати IPsec-тунелі поверх MPLS. До потенційних загроз належать атаки на контрольну площину, зокрема підміна або ін'єкція маршрутних оголошень у BGP/MPLS, компрометація LDP (Label Distribution Protocol) та зловживання механізмами QoS. Для зменшення ризиків застосовуються захист контрольної площини (Control Plane Protection), методи RTBH (Remote Triggered Black Hole), фільтрація маршрутів і уніфікований захист від spoofing (uRPF).

Масштабованість і керованість великих MPLS-мереж потребують додаткових технологічних рішень. Важливим кроком стала поява Segment Routing (SR-MPLS), яка спрощує сигнальну площину, усуває необхідність у RSVP-TE та дозволяє створювати політики маршрутизації з високою гнучкістю. Паралельно розвивається EVPN (Ethernet VPN) — сучасна модель для мульти-tenant середовищ, дата-центрів і гібридних архітектур, що підтримує як L2-, так і L3-сервіси з інтеграцією у хмарні інфраструктури.

Сучасні оператори все частіше комбінують MPLS із технологіями SD-WAN, що дає змогу динамічно балансувати трафік між MPLS-каналами та Інтернет-з'єднаннями залежно від якості й вартості. Такий підхід дозволяє оптимізувати витрати підприємства, зберігаючи високу якість для критичних додатків. Додатково набирає поширення інтеграція MPLS із SASE (Secure Access Service Edge): MPLS використовується як основа для SLA-критичних сервісів, тоді як SASE-шлюзи забезпечують гнучку хмарну безпеку для мобільних і віддалених користувачів.

Перспективи розвитку MPLS пов'язані з поступовим переходом до SRv6 (Segment Routing over IPv6), який розширює гнучкість та масштабованість управління маршрутами, а також із глибшою інтеграцією в гібридні та мультихмарні середовища через cloud interconnect. Зростає роль автоматизації: оркестратори SDN, конфігураційні моделі (NETCONF/YANG) і протоколи PCEP використовуються для

централізованого управління політиками, що значно знижує складність адміністрування.

Таким чином, MPLS є фундаментальною технологією для побудови корпоративних і операторських мереж, яка забезпечує високу передбачуваність і якість сервісів. Проте для досягнення належного рівня інформаційної безпеки необхідна інтеграція із сучасними методами шифрування, моніторингу та автоматизації. У поєднанні з SD-WAN, SASE та SRv6 MPLS зберігає актуальність і стає частиною комплексної моделі захисту та управління мережами майбутнього.

Zero Trust Network Access (ZTNA) — сучасний підхід та набір технологій, що змінює традиційну «периметрову» модель безпеки: замість довіряти всім пристроям у мережі, ZTNA передбачає, що жоден пристрій чи користувач не довіряються за замовчуванням. Доступ надається на основі точкових політик, контексту (прийнятний стан пристрою, місцезнаходження, роль користувача), мінімально необхідних прав (least privilege) і постійної перевірки (continuous authentication/authorization). Технічно ZTNA поєднує елементи мікросегментації, сильну аутентифікацію, адаптивний контроль доступу та видимість у реальному часі (telemetry, EDR/SIEM).

Для практичної реалізації ZTNA часто використовують проксі-моделі, програмно-визначені мережі (SD-WAN/SDN) і інструменти для оркестрації політик; впровадження дає значне підвищення безпеки, але вимагає ретельного планування (карта додатків, критерії довіри, інтеграція з IAM і моніторингом). Сьогодні ZTNA часто реалізують як частину архітектури SASE (Secure Access Service Edge), поєднуючи функції безпеки й мережі в хмарі для глобально розподілених користувачів. Великі організації додають контекстні індикатори ризику (device posture, threat intel, behavior analytics) для адаптивного підвищення або зниження рівня довіри в реальному часі. Інтеграція ZTNA з сучасними IAM-стеками (OIDC, SAML, OPA) і CASB дозволяє контролювати доступ до SaaS-додатків і зменшувати ризики витоку даних. Застосування Continuous Diagnostics & Mitigation (CDM) і автоматичних Playbook-ів робить ZTNA не лише політикою, а й операційним процесом реагування.

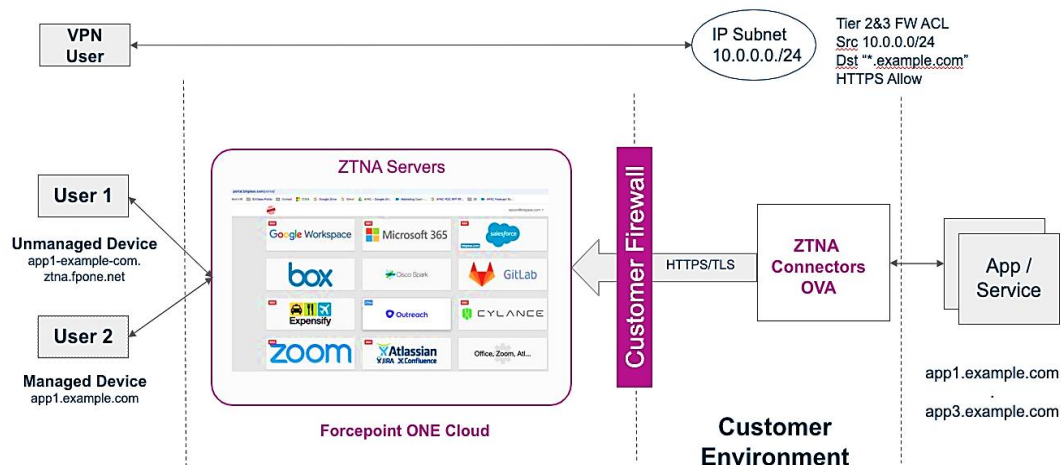


Рис. 3.19. Архітектура доступу Zero Trust Network Access (ZTNA)
Джерело: [1-9, 11-26, 32-48, 60-65, 72-77]

Zero Trust Network Access як сучасна парадигма захисту мереж

Zero Trust Network Access (ZTNA) — це сучасний підхід та набір технологій, що докорінно змінюють традиційну «периметрову» модель безпеки. Якщо класичні архітектури виходили з припущення, що після автентифікації на межі мережі користувачу можна довіряти, то у ZTNA жоден пристрій чи користувач не отримує довіру за замовчуванням. Доступ надається на основі точкових політик та контекстних факторів: відповідності пристрою корпоративним стандартам безпеки, його геолокації, ролі користувача та поточного рівня ризику. Принцип мінімально необхідних прав (least privilege) поєднується з постійною перевіркою (continuous authentication/authorization), що дозволяє адаптивно змінювати рівень доступу.

Технічно ZTNA інтегрує мікросегментацію, багатофакторну автентифікацію, адаптивний контроль доступу та реальновчасну телеметрію (telemetry, EDR/SIEM). Для реалізації архітектури використовуються проксі-моделі, програмно-визначені мережі (SDN/SD-WAN) та засоби оркестрації політик. Важливим аспектом є видимість (observability): система повинна забезпечувати моніторинг трафіку, збір метрик і телеметрії для перевірки коректності політик та їх динамічного коригування. У практичній архітектурі ZTNA часто стає частиною комплексної моделі Secure Access Service Edge (SASE), яка поєднує функції безпеки й мережевої інфраструктури у хмарі для розподілених користувачів.

Впровадження ZTNA базується на моделі зрілості, описаній у NIST SP 800-207 та CISA Zero Trust Maturity Model. На початковому рівні застосовуються лише базові контролю доступу, проте поступово вони

розширюються за рахунок контекстної оцінки ризику, інтеграції з аналітикою поведінки (UEBA) та автоматичної адаптації політик. Ключовим є безперервний розвиток системи, що перетворює ZTNA не на одноразове впровадження, а на постійний процес еволюції.

З погляду інтеграції ZTNA співіснує з іншими стандартами та рамками — ISO/IEC 27001/27002, NIST Cybersecurity Framework (CSF), а також із сучасними IAM-технологіями (OIDC, SAML, OPA). Це дозволяє централізовано управляти правами доступу, контролювати використання SaaS-додатків через CASB та мінімізувати ризик витоку даних. Контекстні індикатори ризику, такі як device posture, threat intelligence або поведінкові профілі користувачів, дозволяють динамічно підвищувати чи знижувати рівень довіри.

У практичному впровадженні ZTNA застосовуються багаторівневі заходи: багатофакторна автентифікація й централізоване IAM для управління правами; наскрізне шифрування; мікросегментація з принципом «least privilege»; централізований моніторинг і регулярний аудит політик; автоматизовані сценарії реагування (playbooks) у поєднанні з SOAR та SIEM. Рекомендується поступовий підхід (maturity model): від пілотних проєктів із критичними додатками до масштабування на всі сервіси підприємства.

Попри переваги, ZTNA має виклики: можливі затримки у доступі через додаткові перевірки, складність інтеграції в legacy-інфраструктури та підвищені вимоги до підготовки персоналу. Однак сучасні тенденції свідчать про швидкий розвиток: інтеграцію ZTNA з AI/ML для прогнозування аномальної поведінки, розширення до Zero Trust Extended (ZTX), що охоплює IoT, OT, 5G та мультимарні середовища, а також використання концепції «Policy-as-Code» і оркестрації через DevSecOps/GitOps-процеси. Таким чином, Zero Trust Network Access стає не лише технологією, а концептуальною парадигмою, яка формує новий стандарт захисту корпоративних мереж. Його поєднання з SASE, SD-WAN та IAM створює комплексну архітектуру, що забезпечує баланс між продуктивністю, масштабованістю та найвищим рівнем кіберзахисту.

У практичній архітектурі ці технології часто комбінуються: MPLS або інтернет-канали можуть забезпечувати транспорт із гарантованим QoS, поверх якого прокладаються IPsec-тунелі для захищеної каналної передачі, а ZTNA — для детального контролю доступу користувачів і сервісів. Рекомендується: (1) застосовувати багатофакторну аутентифікацію та централізоване IAM для управління правами; (2) впроваджувати шифрування на всіх рівнях, де це необхідно; (3) використовувати мікросегментацію і суворі політики «least privilege»; (4) забезпечити

централізований моніторинг, логування та регулярне тестування конфігурацій (пентести, аудит політик). Такий багаторівневий підхід дозволяє одночасно досягти продуктивності, масштабованості та високого рівня захисту інформаційно-мережевих ресурсів підприємства. Додатково важливо впроваджувати «observability» — телеметрію й метрики мережі, щоб політики безпеки можна було перевіряти та автоматично коригувати. Автоматизація оркестрації політик через IaC/CI-CD і інтегровані SOAR/SIEM-копоненти суттєво зменшує час реакції на інциденти. Нарешті, рекомендується поступовий підхід до впровадження (maturity model): тестування на пілотах, навчання персоналу й постійне оновлення політик відповідно до ризик-стадій та бізнес-вимог.

3.6. ВИКЛИКИ БЕЗПЕКИ У ХМАРНИХ ТА SDN-МЕРЕЖАХ

Хмарні мережі забезпечують гнучкість, масштабованість і економічність, однак водночас створюють нові вектори атак. Основним викликом є багатокористувацьке середовище, де ізоляція даних між клієнтами може бути порушена через вразливості в гіпервізорах чи контейнерах. Неправильні налаштування хмарних сервісів часто призводять до витоків конфіденційної інформації, що підтверджується численними інцидентами із відкритими S3-бакетами або відкритими базами даних. Додатковою проблемою є складність контролю за місцем зберігання та обробки даних, що особливо критично для підприємств із жорсткими вимогами до регуляторної відповідності (GDPR, ISO/IEC 27001, HIPAA).

Сучасні постачальники хмарних послуг впроваджують інструменти Data Loss Prevention (DLP) та шифрування «в стані спокою» і «під час передачі» для зменшення ризиків. Все ширше застосовуються технології Confidential Computing, які захищають дані навіть під час обробки у середовищі довірених виконуючих середовищ (TEE). Крім того, активно впроваджуються автоматизовані сервіси для перевірки конфігурацій і дотримання політик безпеки. Це дає змогу швидше виявляти критичні помилки та реагувати на інциденти у хмарі.

Ще одним аспектом безпеки у хмарі є управління ідентичностями та доступом. Традиційна модель захисту периметру вже не працює в умовах динамічного підключення користувачів і сервісів із різних місць. Відсутність чіткої сегментації та політик доступу може дозволити зловмиснику, який отримав компрометовані облікові дані, переміщатися мережею та збирати критичні дані. Сучасні підходи, такі як Zero Trust Network Access (ZTNA), багатобакторна аутентифікація та динамічні політики доступу на основі контексту, покликані знизити ці ризики. До

цього додаються інструменти Cloud Access Security Broker (CASB), що дозволяють централізовано контролювати доступ до SaaS-додатків і запобігати несанкціонованому обміну даними. Популярності набуває концепція Identity as a Service (IDaaS), що інтегрує єдиний вхід (SSO) та умовну автентифікацію на основі ризику. Використання поведінкової аналітики користувачів (UEBA) дозволяє виявляти аномальні дії навіть за наявності правильних облікових даних. Усе це сприяє формуванню адаптивної системи контролю доступу, яка відповідає сучасним динамічним загрозам.

Особливості безпеки у хмарних середовищах

Безпека у хмарних середовищах визначається не лише технічними параметрами, а й організаційною моделлю відповідальності. У практиці використовується Shared Responsibility Model, яка чітко розмежовує обов'язки: постачальник хмарних послуг відповідає за захист інфраструктури (фізичні дата-центри, віртуалізаційний шар, базові сервіси), тоді як користувач зобов'язаний захищати власні дані, керувати доступами, налаштовувати безпекові політики й контролювати використання додатків. Недотримання цього принципу часто стає причиною інцидентів, коли клієнти помилково покладаються на провайдера у сферах, які належать до їхньої зони контролю.

Окремим викликом є мультихмарні та гібридні архітектури, де одночасно використовується кілька провайдерів. Це ускладнює централізоване управління безпекою, адже різні платформи мають відмінні механізми контролю доступу, моніторингу й комплаєнсу. Для зниження цих ризиків активно впроваджуються Cloud Security Posture Management (CSPM) та Cloud Workload Protection Platforms (CWPP), які забезпечують автоматичне виявлення неправильних конфігурацій, контроль відповідності стандартам і реагування на інциденти.

Важливим аспектом безпеки у хмарі є регуляторна відповідність. Підприємства, що працюють із персональними чи фінансовими даними, зобов'язані виконувати вимоги GDPR, HIPAA, PCI DSS, ISO/IEC 27017/27018. Це передбачає контроль за місцем зберігання інформації, криптографічний захист даних «у спокої» та «під час передачі», створення журналів аудиту та забезпечення доказової бази для розслідування інцидентів.

З огляду на динаміку загроз, сучасні хмарні провайдери впроваджують Confidential Computing, що дозволяє захищати дані навіть під час обробки у довірених виконуючих середовищах (TEE). Додатково набуває поширення поведінкова аналітика (UEBA) та машинне навчання для виявлення аномальної активності в режимі реального часу.

Перспективним напрямом є інтеграція принципів Zero Trust у хмарні середовища, що забезпечує контроль доступу на основі контексту й постійну перевірку довіри. У майбутньому актуальним стане впровадження post-quantum cryptography (PQC) для захисту довготривалих ключів та критичних каналів зв'язку у хмарній інфраструктурі.

Безпека у SDN-мережах

У SDN (Software-Defined Networking) основний виклик безпеки полягає у централізованій природі управління. Контролер SDN є критичною точкою, компрометація якої може відкрити доступ до всієї мережевої інфраструктури. Канали між контролером і комутаторами також можуть бути атаковані, якщо не використовуються протоколи шифрування та аутентифікації. Водночас програмні інтерфейси (API), через які адміністратори керують мережею, стають мішенню атак типу injection або DoS, що здатні паралізувати роботу всієї мережі.

Модель загроз для SDN охоплює низку специфічних векторів: підміну маршрутів, атаки на Southbound API (наприклад, OpenFlow), ін'єкції в Northbound API, DoS-атаки на контролер та внутрішні загрози від скомпрометованих адміністраторів чи додатків. Саме формалізація цих ризиків дозволяє чіткіше визначати вимоги до безпеки, що включають сегментацію доступу, аудит взаємодії між рівнями та контроль цілісності програмних модулів.

Сучасні дослідження рекомендують застосовувати моделі розподіленого контролю з резервуванням, щоб уникнути єдиної точки відмови. Використання протоколів TLS 1.3 та взаємної аутентифікації між контролером і пристроями значно знижує ризики перехоплення чи підробки команд. Активно впроваджуються підходи на основі машинного навчання для виявлення аномалій у керуючому трафіку, а також інтегровані засоби валідації API, які перевіряють запити перед їхнім виконанням.

Окрему увагу слід приділяти регуляторним і стандартним аспектам. Оскільки SDN все частіше використовується у критичній інфраструктурі та хмарних середовищах, застосовуються вимоги ISO/IEC 27001, NIST Cybersecurity Framework та ETSI NFV Security. Вони передбачають централізоване управління ключами, аудит усіх змін у конфігурації, контроль доступу на основі ролей (RBAC/ABAC) і регулярне тестування політик.

Зростає значення концепції Zero Trust для SDN, яка обмежує довіру навіть усередині самої інфраструктури й мінімізує наслідки компрометації окремих вузлів. З'являються рішення з використанням блокчейн-технологій для забезпечення прозорості змін конфігурацій і неможливості їх підробки. У виробничих середовищах набувають поширення контейнеризовані

контролери, що дозволяють швидко оновлювати програмне забезпечення без зупинки мережі. Додатково розробляються політики Security-as-Code, інтегровані у CI/CD-конвеєри, які забезпечують автоматичну перевірку та застосування правил безпеки.

Багаторівнева архітектура SDN складається з трьох основних рівнів. На рівні додатків (Application Layer) формуються політики безпеки, пріоритети трафіку та вимоги до QoS; він взаємодіє з контролером через Northbound API, що дозволяє інтегрувати системи аналітики, моніторингу, виявлення аномалій і автоматизованого реагування. Контрольний рівень (Control Layer), реалізований SDN-контролером, виступає «мозком» мережі: він отримує вимоги, аналізує стан інфраструктури та формує правила маршрутизації й захисту. Сучасні контролери мають розподілену архітектуру, що підвищує відмовостійкість і дозволяє застосовувати алгоритми машинного навчання для прогнозування аномалій. Рівень пересилання даних (Data Plane) складається з мережевих пристроїв, які виконують інструкції контролера через стандартизовані протоколи (OpenFlow, NETCONF, P4), забезпечуючи узгодженість і високу швидкодію.

Багаторівнева модель SDN забезпечує централізований контроль, швидке масштабування й автоматизацію, але водночас створює виклик: контролер стає критичною точкою уразливості. Тому захист має ґрунтуватися на комплексному підході: сегментації, автентифікації, шифруванні, моніторингу та постійних аудитах.

На рівні додатків (Application Layer) розміщуються бізнес-сервіси та прикладні програми, які формують вимоги до мережі: політики безпеки, пріоритети трафіку, потреби в QoS чи балансуванні навантаження. Цей рівень взаємодіє з контролером через відкриті інтерфейси (Northbound API), що дозволяє додаткам напряму впливати на поведінку мережі. На рівні додатків (Application Layer) розміщуються бізнес-сервіси та прикладні програми, які формують вимоги до мережі: політики безпеки, пріоритети трафіку, потреби в QoS чи балансуванні навантаження. Цей рівень взаємодіє з контролером через відкриті інтерфейси (Northbound API), що дозволяє додаткам напряму впливати на поведінку мережі. Додатки цього рівня можуть включати системи аналітики, моніторингу, виявлення аномалій і автоматизованого реагування на інциденти. Завдяки цьому підприємства отримують можливість оперативно адаптувати мережу до змін бізнес-процесів і загроз. Також цей рівень забезпечує інтеграцію з хмарними сервісами та оркестраторами контейнерів, що спрощує управління гібридною інфраструктурою. Використання відкритих API сприяє розвитку екосистеми додатків від сторонніх розробників, що підвищує гнучкість і

масштабованість. У результаті Application Layer стає ключовим елементом для реалізації інновацій, автоматизації та впровадження нових бізнес-моделей у сучасних мережах.

Контрольний рівень (Control Layer) реалізується за допомогою SDN-контролера, який виступає своєрідним «мозком» мережі. Контролер отримує вимоги від додатків, аналізує поточний стан інфраструктури та формує правила, що визначають маршрутизацію і захист трафіку. Він забезпечує централізовану логіку управління, адаптацію до змін мережевих умов і інтеграцію з системами моніторингу та кіберзахисту. Контрольний рівень (Control Layer) реалізується за допомогою SDN-контролера, який виступає своєрідним «мозком» мережі. Контролер отримує вимоги від додатків, аналізує поточний стан інфраструктури та формує правила, що визначають маршрутизацію і захист трафіку. Він забезпечує централізовану логіку управління, адаптацію до змін мережевих умов і інтеграцію з системами моніторингу та кіберзахисту. Сучасні SDN-контролери підтримують розподілену архітектуру для підвищення стійкості та відмовостійкості мережі. Вони здатні автоматично оптимізувати маршрути залежно від навантаження і забезпечувати балансування трафіку між різними каналами. Крім того, контролери інтегруються з інструментами машинного навчання для прогнозування аномалій та кіберзагроз. У результаті цей рівень стає ключовим елементом інтелектуального управління, що дозволяє швидко реагувати на інциденти та підвищувати ефективність роботи всієї мережі.

Рівень пересилання даних (Data Plane або Forwarding Layer) складається з мережевих пристроїв, які фізично здійснюють передачу пакетів. Ці пристрої не приймають рішень самостійно, а виконують інструкції контролера через стандартизовані протоколи, такі як OpenFlow, NETCONF чи P4. Завдяки цьому досягається узгодженість у роботі всієї мережі та зменшується складність адміністрування. Рівень пересилання даних (Data Plane або Forwarding Layer) складається з мережевих пристроїв, які фізично здійснюють передачу пакетів. Ці пристрої не приймають рішень самостійно, а виконують інструкції контролера через стандартизовані протоколи, такі як OpenFlow, NETCONF чи P4. Завдяки цьому досягається узгодженість у роботі всієї мережі та зменшується складність адміністрування. Додатково цей рівень забезпечує високу швидкодію, оскільки вся інтелектуальна обробка винесена на контрольний рівень. У сучасних SDN-мережах пристрої data plane можуть бути як апаратними, так і віртуальними, що спрощує інтеграцію з хмарними сервісами та контейнерними середовищами. Також реалізується підтримка програмного визначення політик QoS, пріоритизації трафіку та вбудованих механізмів

безпеки. Це робить data plane критично важливим елементом для стабільності та ефективності всієї мережевої архітектури.

Багаторівнева архітектура SDN забезпечує низку важливих переваг. По-перше, вона дозволяє централізовано керувати всією мережею, знижуючи витрати на обслуговування та спрощуючи конфігурацію. По-друге, вона створює умови для швидкого масштабування та автоматизації операцій, що особливо важливо для дата-центрів і хмарних сервісів. По-третє, SDN відкриває можливість динамічно реалізовувати політики безпеки та інтегрувати новітні технології, включно з Zero Trust, машинним навчанням для виявлення аномалій та оркестрацією через CI/CD.

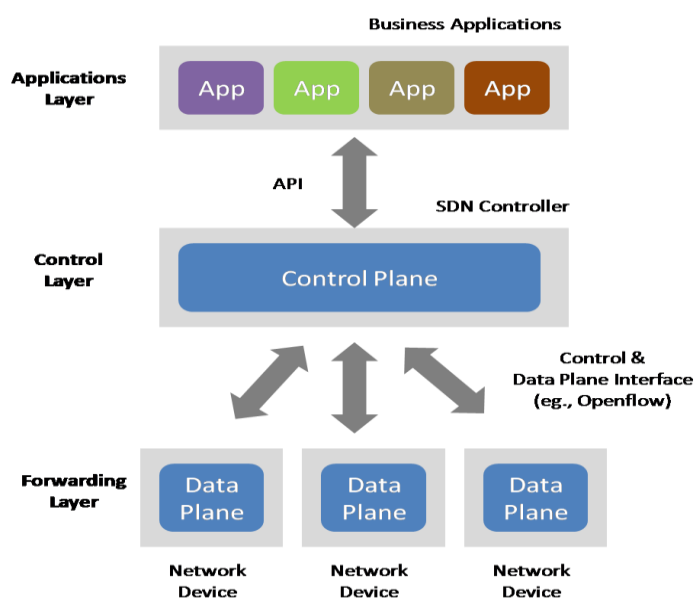


Рис. 3.20. Багаторівнева архітектура SDN (Software-Defined Networking)
Джерело: [1-9, 11-26, 32-48, 60-65, 72-77]

Разом із тим, така архітектура має і виклики. Центральний контролер стає критичною точкою, компрометація якої може поставити під загрозу всю мережу. Канали зв'язку між рівнями повинні бути надійно захищені за допомогою сучасних криптографічних протоколів. Також зростає важливість контролю доступу до API, оскільки саме вони визначають взаємодію між додатками, контролером і пристроями. Саме тому безпека в SDN має базуватися на комплексному підході: сегментації, автентифікації, шифруванні та постійному моніторингу.

У підсумку багаторівнева архітектура SDN є сучасним підходом до побудови мереж, що поєднує гнучкість, централізований контроль і можливість швидкої адаптації до нових умов. Вона відкриває шлях до побудови інтелектуальних, автоматизованих і безпечних мережевих

середовищ, здатних відповідати викликам цифрової трансформації та динамічного розвитку кіберзагроз. Сьогодні її все частіше поєднують із концепціями Zero Trust та SASE для комплексного захисту даних і сервісів у гібридних інфраструктурах. Інтеграція з технологіями штучного інтелекту дозволяє мережам прогнозувати навантаження, виявляти аномалії та автоматично реагувати на загрози. Важливим напрямом розвитку є впровадження квантово-стійкої криптографії для захисту каналів управління. У результаті SDN стає основою для формування адаптивних і стійких до атак мереж майбутнього.

Додатковим викликом у SDN є забезпечення довіри між різнорідними компонентами та сервісами. В умовах мультихмарності та використання віртуалізованих функцій (NFV) особливо важливим стає контроль за цілісністю та автентичністю програмних модулів. Вразливості в оркестраційних системах чи зловмисні оновлення можуть призвести до втрати керованості мережею. Для подолання цих загроз необхідні механізми сегментації, мікросервісної ізоляції, криптографічного захисту каналів зв'язку та багаторівневого моніторингу. Новітнім підходом є інтеграція Zero Trust принципів у SDN для обмеження довіри навіть між внутрішніми сервісами. Використання блокчейн-технологій для перевірки автентичності та історії змін конфігурацій стає перспективним напрямом досліджень. Набуває популярності підхід Security-as-Code, коли політики безпеки впроваджуються й автоматично перевіряються через CI/CD конвеєри. Це дозволяє мінімізувати людські помилки й швидко реагувати на нові вразливості у середовищі SDN/NFV.

У перспективі особливу роль відіграватимуть квантово-стійкі алгоритми шифрування, здатні забезпечити довготривалий захист керуючих каналів, а також інтеграція з концепціями Zero Trust та SASE для формування самозахищених і адаптивних мереж. Використання DevSecOps і Policy-as-Code перетворює безпеку SDN на невід'ємну частину життєвого циклу мережевої інфраструктури. У результаті SDN стає основою побудови інтелектуальних, стійких до атак мереж майбутнього, здатних протидіяти динамічним загрозам у реальному часі.

3.7. ВРАЗЛИВОСТІ ТА ЕКСПЛОЙТИ: ВІД ПОМИЛОК КОНФІГУРАЦІЇ ДО ZERO-DAY АТАК

Вразливості в комп'ютерних мережах та пов'язані з ними експлойти становлять основну рушійну силу успішних атак, тому їх розуміння і системне управління є ключовими елементами захисту; ці вразливості умовно поділяються на помилки конфігурації, програмні дефекти й так звані

zero-day, кожна з яких має свої механізми виникнення й особливості експлуатації.

Під вразливостями розуміються будь-які слабкі місця в архітектурі, налаштуванні або програмному забезпеченні мережі, які можуть бути використані для порушення конфіденційності, цілісності чи доступності даних. Вони включають як помилки конфігурації (відкриті порти, дефолтні облікові записи, слабкі політики доступу), так і програмні баги (переповнення буфера, ін'єкції, уразливості у стеку протоколів). Окрему категорію становлять zero-day уразливості — ті, що ще не мають офіційних патчів і є особливо цінними для кібератак, оскільки системи захисту не можуть виявляти їх сигнатурно.

Поняття експлойтів описує конкретні програмні або скриптові засоби, за допомогою яких зловмисники використовують уразливості: від скриптів для SQL-ін'єкцій чи ARP-spoofing до складних багаторівневих APT-інструментів для віддаленого виконання коду (RCE). Важливо, що експлойти класифікуються за вектором атаки (локальні чи віддалені), за результатом (DoS, підвищення привілеїв, отримання доступу), а також за рівнем автоматизації (від простих proof-of-concept до комерційних exploit kit).

Помилки конфігурації

Найпоширеніший клас вразливостей, що виникає через некоректне налаштування мережевих пристроїв, серверів або програмного забезпечення. Прикладами є відкриті порти без потреби, відсутність сегментації трафіку, використання слабких або дефолтних паролів, неправильні ACL (Access Control Lists). Такі помилки часто стають причиною компрометації корпоративних мереж, оскільки дають змогу зловмиснику легко обійти базові механізми контролю доступу.

Помилки конфігурації — це найпростіший і тому найпоширеніший вектор проникнення. Неправильно налаштовані служби, відкриті непотрібні порти, дефолтні або слабкі облікові дані, незакриті адміністративні інтерфейси, відсутність сегментації мережі й некоректні ACL створюють «шлюзи» для початкового доступу й подальшого латерального руху.

Рисунок ілюструє типовий сценарій компрометації через помилку конфігурації й складається з простих символів, що передають логіку атаки. Зліва зображено фігуру атакувальника в капюшоні — джерело загрози; дві пунктирні червоні стрілки показують мережеві потоки від нього в бік внутрішньої інфраструктури. Посередині стоїть умовний «файрвол»/бар'єр (кляцаний або цегляний стовп), але стрілки проходять далі — це візуально показує, що захисний периметр налаштований невірно або пропускає небажаний трафік. Справа — іконка замочка, що позначає захищений

ресурс або сервіс, до якого атакувальник отримує доступ. Загалом діаграма демонструє просту думку: через неправильно налаштовані правила доступу (відкриті порти, слабкі ACL, публічні адмін-інтерфейси тощо) зловмисник може «пробити» мережевий бар'єр і дістатися до внутрішніх ресурсів. Структура рисунка також натякає на послідовність дій — від зовнішньої розвідки й сканування (стрілки зліва) через використання конфігураційного шлюзу до отримання несанкціонованого доступу (замок справа). Таким чином, цей рисунок служить наочною інфографікою ризику misconfiguration: навіть наявність периферійних засобів захисту не дає гарантії без правильно налаштованих правил та політик доступу.

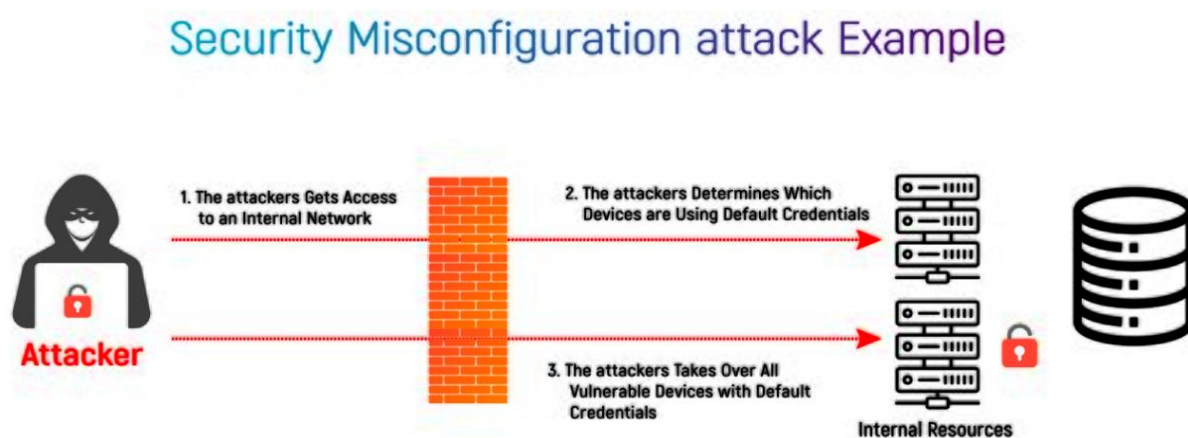


Рис. 3.21. Приклад атаки через помилку конфігурації
Джерело: [1-9, 11-26, 32-48, 60-65, 72-77]

Діаграма «Типові помилки конфігурації в хмарних середовищах» показує центральну хмарну іконку як ядро проблеми, навколо якої розташований круговий стрічковий графік із кольоровими сегментами — кожен сегмент відповідає конкретному типу помилки конфігурації: незахищене зберігання конфіденційних даних (коричневий), надмірні права й політики доступу (зелений), неналежні механізми контролю доступу (синій), помилково налаштовані правила фаєрвола (жовтий), недостатня мережева сегментація (блакитний), відсутність адекватного моніторингу й логування (сірий), використання застарілого ПЗ і бібліотек (персиковий) та необмежені вхідні/вихідні порти (фіолетовий). Кругова композиція підкреслює, що ці проблеми взаємопов'язані й часто співіснують у циклічних векторах ризику: одна вразливість (наприклад, відкритий порт) може полегшити експлуатацію іншої (наприклад, незашифроване сховище), а недбале логування ускладнює виявлення інцидентів. Легенда праворуч дає коротку маркування кожного кольору для швидкої орієнтації, а візуальна форма — зручна для використання на слайді, де можна додати короткі

пояснення, пріоритети виправлення або посилення на відповідні контролі (hardening, IAM, WAF, моніторинг). Діаграма служить наочною відправною точкою для обговорення пріоритетів безпеки в хмарі і дозволяє аудиторії швидко побачити спектр типових misconfiguration, їх взаємозв'язки та зони, що потребують першочергової уваги.

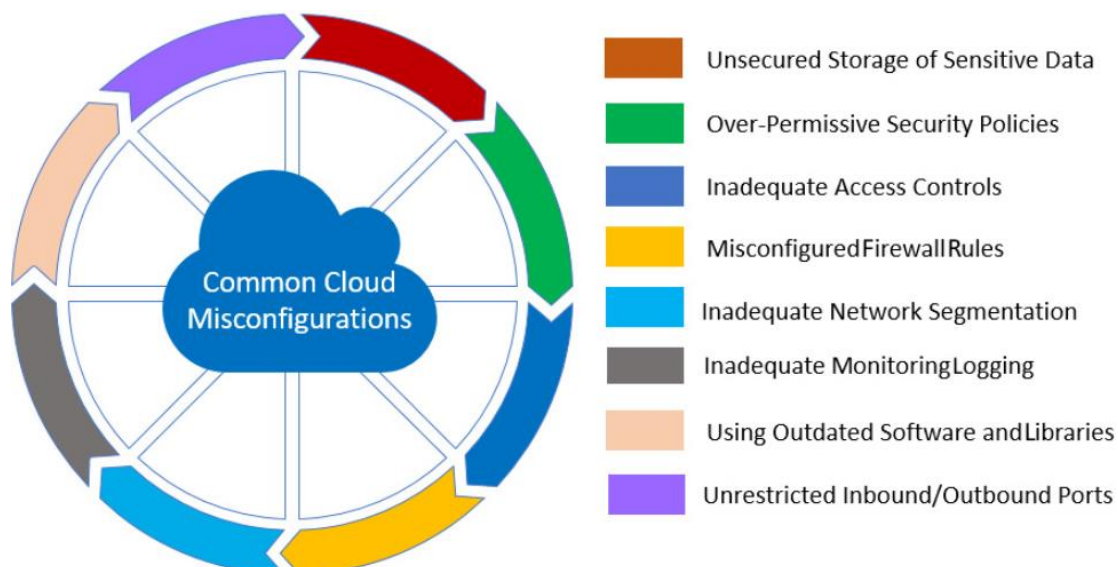


Рис. 3.22. Типові помилки конфігурації в хмарних середовищах
Джерело: [1-9, 11-26, 32-48, 60-65, 72-77]

Типовий алгоритм дій атакувальника при експлуатації помилок конфігурації має кілька послідовних кроків. Спершу відбувається розвідка — збирання публічної й внутрішньої інформації про ціль, визначення відкритих портів і версій сервісів. Далі — сканування, під час якого виявляються відкриті порти, служби і відомі вразливості. Наступний крок — верифікація, що включає перевірку доступності адміністративних інтерфейсів і пошук дефолтних або слабких паролів. Потім настає експлуатація знайдених слабких місць для входу або виконання коду. Завершальні етапи — закріплення й латеральний рух (отримання облікових даних, розгортання бекдорів, підйом привілеїв, переміщення по мережі) та приховання слідів.

Для кожного кроку існують відпрацьовані інструменти. На етапі розвідки й сканування застосовуються Nmap, Masscan, Shodan чи Zoomeye (для зовнішніх поверхонь). Для виявлення уразливих сервісів і версій використовуються Nessus, OpenVAS. Перевірка дефолтних облікових даних та брутфорс виконуються за допомогою Hydra, Medusa, Ncrack. Для експлуатації застосовують Metasploit (модулі RCE, експлойти для

конкретних CVE), спеціалізовані exploit-кити і публічні PoC. Для перехоплення локальних мережеских обмінів і хешів — Responder, Ettercap. Для підйому привілеїв і роботи з обліковими даними в пам'яті — Mimikatz. Для картографії доменних зв'язків і підбору шляхів латерального руху в Active Directory — BloodHound. Для автоматизації сценаріїв компрометації застосовують набори скриптів на Python/PowerShell або C2-framework (Cobalt Strike та ін.). Усі ці інструменти роблять помилки конфігурації особливо небезпечними: навіть одна відкрита консоль управління або слабкий пароль можуть призвести до повного зламу периметра.

Nmap

Nmap є класичним інструментом для сканування портів та визначення служб і їхніх версій (service/version detection). Він дозволяє зібрати карту мережевої поверхні, виявити відкриті порти, RDP чи SSH-сервіси, адміністративні інтерфейси та зібрати дані про версії ПЗ для підбору потенційних експлойтів. Використання зловмисниками полягає у пошуку “вхідних дверей” для подальшої атаки. Основними індикаторами є велика кількість TCP/UDP-з'єднань з одного джерела за короткий час, характерні послідовності скан-пакетів із SYN/ACK флагами, а також перебір портів у незвичній послідовності. Захист включає фільтрацію й лімітування трафіку на фаєрволах, застосування rate-limiting на прикордонних пристроях, блокування IP-адрес джерел сканування, моніторинг зовнішніх ASM-сервісів (Shodan-watch) та налаштування IDS/NDR правил на виявлення port-sweep активності.

Masscan

Masscan відрізняється від Nmap швидкістю та орієнтацією на масові широкомасштабні сканування інтернет-простору. Його зловмисники використовують для швидкого пошуку великої кількості хостів з відкритими портами, як правило, на етапі первинної розвідки. Індикаторами є величезна кількість однотипних скан-пакетів за короткий проміжок часу, нестандартні інтервали між з'єднаннями та високий рівень однофазних SYN-запитів. Для протидії застосовуються rate-limiting на периметрі, чорні та сірі списки IP, honeypot чи black-hole механізми для відстеження сканерів, а також моніторинг аномалій у трафіку засобами NDR.

Shodan та Zoomeye

Shodan і Zoomeye є інструментами OSINT-пошуку, які індексують публічно доступні сервіси в інтернеті та дозволяють здійснювати пошук за баннерами, портами, географічним розташуванням. Зловмисники користуються цими системами для виявлення публічних адмін-інтерфейсів, незахищених баз даних, камер чи IoT-пристроїв. Самі по собі ці інструменти не створюють трафіку у вашій мережі, але індикатором ризику може стати

поява цілеспрямованих атак після індексування. Захист полягає у приховуванні або закритті адміністративних інтерфейсів, застосуванні аутентифікації та VPN для адміндоступу, моніторингу exposed-активів через ASM або Threat Intelligence, а також обмеженні публічно доступної інформації у WHOIS та SSL-сертифікатах.

Nessus та OpenVAS

Nessus і OpenVAS — це сканери вразливостей, які автоматично перевіряють хости на наявність відомих CVE та помилок конфігурації. Хоча вони зазвичай застосовуються фахівцями із захисту, зловмисники також можуть використати подібні результати для визначення цілей. Індикатори включають часті повторювані запити до різних служб, характерні HTTP/SMB-патерни або «сліди» Nessus-like сканів. Захист полягає у регулярному власному скануванні, інтеграції результатів у процес patch-management, закритті небезпечних служб, дотриманні SLA на виправлення критичних CVE та моніторингу підозрілих спроб зовнішнього сканування.

Hydra, Medusa та Ncrack

Ці інструменти автоматизують брутфорс і credential stuffing, дозволяючи перевіряти дефолтні чи злиті облікові дані в SSH, RDP, FTP або веб-формах. Їхня експлуатація базується на масовому підборі паролів. Основні індикатори включають аномально високу кількість невдалих логінів з одного або з багатьох IP-адрес, однотипні помилки авторизації та сплески failed-login подій. Захист включає впровадження обов'язкової MFA, блокування IP чи акаунтів після певної кількості невдалих спроб, rate-limiting, captcha для веб-форм, кореляцію аномалій у SIEM/UEBA та застосування PAM для контролю привілеїв.

Metasploit, exploit-кити та публічні PoC

Metasploit є платформою для тестування експлойтів, яка часто використовується і атакувальниками, і фахівцями з кібербезпеки. Зловмисники застосовують його для віддаленого виконання коду та розгортання бекдорів після виявлення уразливості. Індикатори включають нетипові завантаження бінарних файлів, підозрілі payloads, а також незвичну поведінку процесів та мережевих сесій на нестандартних портах. Захист полягає у швидкому патчуванні, закритті зайвих сервісів, застосуванні WAF для веб-додатків, EDR для виявлення підозрілих виконань і моніторингу нетипових з'єднань.

Responder та Ettercap

Responder та Ettercap призначені для проведення атак типу MITM і перехоплення локального трафіку, зокрема шляхом ARP-спуфінгу чи підміни відповідей протоколів LLMNR/NBT-NS. Атакувальники використовують їх для викрадення хешів автентифікації та виконання relay-

атак. Основні індикатори: множинні аномальні ARP-відповіді, незвичні DNS/LLMNR-запити та раптові сплески аутентифікаційних хешів. Захист включає відключення LLMNR/NBT-NS, увімкнення SMB signing, сегментацію VLAN, застосування 802.1X, моніторинг ARP-таблиць та використання NDR для виявлення MITM-атак.

Mimikatz

Mimikatz — це інструмент, що дозволяє вилучати паролі, хеші та токени з пам'яті Windows (LSASS). Зловмисники використовують його для крадіжки облікових даних і подальшого підвищення привілеїв. Основними індикаторами є доступ сторонніх процесів до LSASS, запуск підозрілих утиліт та процесів, що читають пам'ять інших програм. Захист базується на EDR із credential dumping prevention, включенні LSA protection, мінімізації кількості адміністраторських акаунтів, контролі виконання скриптів, моніторингу використання подібних утиліт та застосуванні MFA для критичних систем.

BloodHound

BloodHound збирає дані про зв'язки й права доступу в Active Directory, формуючи карту можливих шляхів атаки та отримання доменних привілеїв. Атакувальники застосовують його для аналізу слабких конфігурацій та планування латерального руху. Індикатори — це незвичні запити до AD, експорт ACL чи метаданих, підозріло великі обсяги даних, зібраних у короткий час. Захист включає аудит і обмеження делегувань, регулярні рев'ю прав, застосування tiered-administration, моніторинг підозрілих AD-запитів та hardening контролерів домену.

C2-framework (Cobalt Strike та інші)

Фреймворки команд-і-контролю, такі як Cobalt Strike, забезпечують атакувальникам стабільне управління скомпрометованими хостами, виконання команд, завантаження модулів, lateral movement і ексфільтрацію даних. Індикаторами є довготривалі HTTPS/HTTP-сесії до нетипових доменів, регулярні «маячки», зашифровані payloads та аномальні процеси, що ініціюють зовнішні з'єднання. Захист передбачає моніторинг підозрілих доменів і TLS-сесій, контроль DNS-запитів (виявлення DGA), EDR для виявлення living-off-the-land технік, сегментацію мережі для локалізації інцидентів та автоматизацію реагування через SOAR.

Захист від атак потребує багатошарового підходу і одночасного поєднання технічних, процесних і організаційних заходів.

По-перше, необхідна повна інвентаризація активів і сервісів (CMDB), регулярне сканування на наявність відкритих портів і незадокументованих сервісів, а також автоматичне виявлення змін у конфігурації.

По-друге, потрібно впровадити політики жорсткої конфігурації: закриття непотрібних портів та служб, видалення або деактивація дефолтних облікових записів, застосування CIS-бенчмарків чи власних конфігураційних профілів, автоматичне hardening образів через Infrastructure as Code (Ansible, Terraform) і конфігураційний менеджмент.

По-третє, принцип найменших привілеїв та застосування PAM-рішень (Privileged Access Management), багатофакторної автентифікації для адміністративного доступу й ротації секретів з використанням секрет-менеджерів (HashiCorp Vault, Azure Key Vault) значно ускладнюють використання вкрадених облікових даних.

По-четверте, мережеві контрзаходи — сегментація (VLAN, VRF), мікросегментація для хмарних середовищ, жорсткі ACL на мережевих пристроях, обмеження RDP/SSH доступу через jump-hosts і VPN із подвійною аутентифікацією — перебивають шляхи латерального руху.

По-п'яте, виявлення і реагування: розгортання EDR для кінцевих точок, NDR/IDS/IPS для мережевого трафіку, централізований логінг у SIEM з кореляцією аномалій (висока частота невдалих входів, підозрілі підключення до адміністративних портів, раптові зміни конфігурацій) і автоматизовані playbooks у SOAR для швидкої ізоляції.

Додаткові специфічні заходи включають: відключення LLMNR/NBT-NS і включення SMB signing для захисту від Responder-типу атак; застосування WAF для захисту веб-інтерфейсів; обмеження публічних адміністративних інтерфейсів через VPN/Zero Trust; регулярні pentest і purple-team вправи для виявлення реальних конфігураційних помилок; впровадження процесу patch- і config-management з SLA на критичні виправлення.

Нарешті, важливі процесні елементи — навчання адміністраторів і користувачів, рев'ю конфігурацій перед деплоєм, аудит журналів, процедури emergency-patching та ретельне документування інцидентів і lessons-learned.

Поєднання виявлення відкритих поверхонь (сканування), превентивного закриття вразливостей (hardening, MFA, сегментація) і швидкого реагування (EDR/SOAR/SIEM) робить помилки конфігурації набагато менш корисними для атакувальника й істотно підвищує загальну стійкість мережі.

Програмні вразливості

Ця категорія охоплює недоліки у коді протоколів чи додатків, які можуть бути використані через експлойти. Серед класичних прикладів — переповнення буфера, уразливості у веб-серверах (Apache, IIS), помилки в

реалізаціях SSL/TLS, DNS Cache Poisoning чи уразливості маршрутизаторів. Програмні вразливості здебільшого мають CVE-ідентифікатори, а їхня критичність оцінюється за шкалою CVSS.

Програмні вразливості — це критичний клас недоліків, які виникають унаслідок помилок у коді додатків, бібліотек або мережових протоколів. Їхня експлуатація зловмисниками дозволяє отримати віддалений контроль над системою, виконати довільний код, обійти механізми автентифікації чи отримати доступ до конфіденційної інформації.

Рисунок ілюструє ключові підходи з доповіді NIST (NISTIR-8151) до зменшення вразливостей у програмному забезпеченні: він поданий у вигляді вертикальних колонок-секцій, кожна з яких представляє окрему стратегію — формальні методи, системний рівень безпеки, поєднаний (additive) аналіз інструментів, доменно-специфічні фреймворки та moving-target / software diversity. Кожна секція містить короткий опис і метафоричну іконографіку (наприклад, символи верифікації, модульної архітектури, співпраці інструментів, спеціалізованих мов або «рухомого» захисту), що підкреслює суть підходу: від математичних доказів коректності й модульності систем до автоматичного різноманіття реалізацій і динамічної мінливості платформи. Візуально композиція наголошує на комплексності проблеми й необхідності комбінації методів — жоден підхід сам по собі не дає гарантій, натомість їхнє поєднання підвищує загальну стійкість ПЗ. Діаграма підходить як ілюстрація для презентації або розділу політики безпечної розробки: вона допомагає аудиторії швидко побачити спектр практичних заходів — від культурних/процесних змін (вибір мови й фреймворка) до технічних механізмів (формальна верифікація, модульність, інтеграція аналізаторів). Для доповнення слайду корисно додати короткі приклади для кожної колонки (інструменти, практики, очікувані результати) і посилання на NISTIR-8151 як джерело.

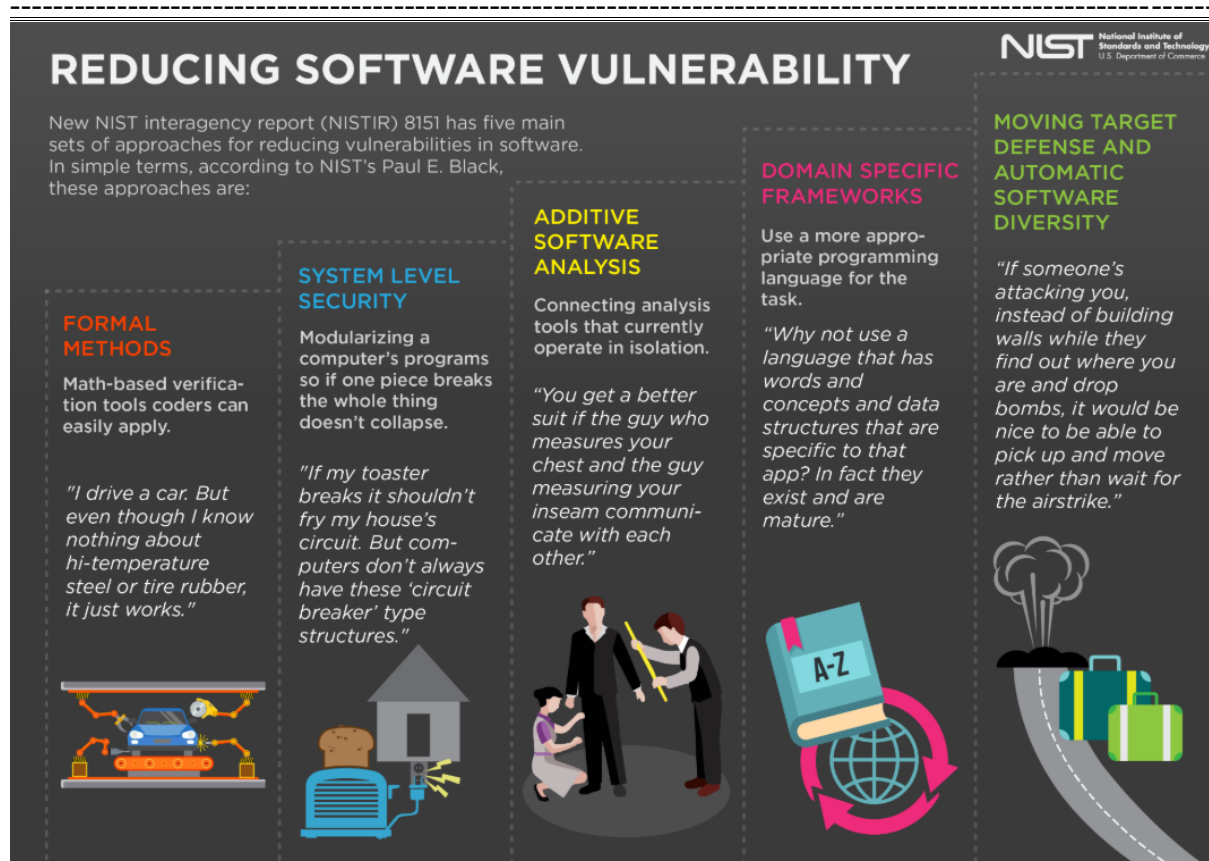


Рис. 3.23. П'ять підходів NIST до зменшення вразливостей у програмному забезпеченні

Джерело: [1-9, 11-26, 32-48, 60-65, 72-77]

Переповнення буфера

Переповнення буфера є одним із найстаріших та водночас найнебезпечніших типів програмних вразливостей. Воно виникає тоді, коли програма не контролює розмір вхідних даних, що записуються у виділений буфер пам'яті. Якщо дані перевищують межі буфера, відбувається їхній вихід за межі виділеної області, унаслідок чого можуть бути перезаписані сусідні ділянки пам'яті, включаючи вказівники на виконуваний код. Це дозволяє зловмиснику вставити власний шкідливий код у стек або купу та змусити процесор виконати його замість легітимної програми. Алгоритм атаки зазвичай включає підбір спеціально сформованого payload, який не тільки переповнює буфер, а й містить shellcode для відкриття віддаленого доступу чи підвищення привілеїв. Для реалізації подібних атак активно застосовуються публічні експлойти з бази Exploit-DB, готові модулі Metasploit, а також індивідуальні PoC на мові C чи Python. Захист від таких атак передбачає впровадження технологій ASLR та DEP, компіляцію коду з опціями безпечної роботи з пам'яттю, статичний і динамічний аналіз програм, а також регулярне оновлення бібліотек і середовищ виконання.

Вразливості у веб-серверах (Apache, IIS)

Веб-сервери є однією з найпривабливіших цілей атак, оскільки забезпечують критичні сервіси доступу до ресурсів підприємства. Вразливості у веб-серверах часто виникають через помилки в обробці HTTP-запитів, некоректні модулі чи плагіни, слабкі конфігурації та відсутність належної автентифікації. Наприклад, помилки у версіях Apache чи IIS дозволяли виконати довільний код на сервері через спеціально сформовані запити, використати directory traversal для доступу до конфіденційних файлів чи здійснити denial-of-service атаки. Алгоритм експлуатації зазвичай полягає у відправці спеціально сформованого HTTP-запиту, що викликає некоректну поведінку веб-сервера, і подальшому завантаженні або виконанні шкідливого коду. Для цього зловмисники застосовують Burp Suite, OWASP ZAP, sqlmap або готові Metasploit-модулі. Захист включає застосування веб-фаєрволів (WAF), регулярні оновлення серверного ПЗ, відключення непотрібних модулів і сервісів, моніторинг логів доступу, а також сегментацію веб-серверів у DMZ для обмеження впливу успішної атаки.

Помилки в реалізаціях SSL/TLS

Криптографічні протоколи SSL/TLS, які використовуються для захисту передавання даних у мережі, теж мають власні вразливості, що у свій час ставали глобальними проблемами безпеки. Одним із відомих прикладів є Heartbleed у бібліотеці OpenSSL, що дозволяв читати випадкові ділянки пам'яті сервера та отримувати секретні ключі чи облікові дані. Інші вразливості, такі як POODLE, експлуатували слабкі алгоритми шифрування та дозволяли знизити рівень захисту до небезпечних версій протоколу. Атакувальники зазвичай починають із визначення версії SSL/TLS, що використовується на сервері, і перевіряють її на наявність відомих CVE. Далі застосовуються готові PoC або інструменти, як-от testssl.sh чи Metasploit-модулі для Heartbleed. Захист базується на відключенні застарілих версій SSL (2.0, 3.0) і слабких шифрів, регулярному оновленні бібліотек OpenSSL та впровадженні TLS 1.3, а також моніторингу трафіку на предмет підозрілих downgrade-атак.

DNS Cache Poisoning

DNS Cache Poisoning належить до атак, що використовують слабкі місця у реалізації DNS-серверів та механізмах кешування. Принцип дії полягає у підміні кешованих записів у DNS-сервері жертви так, щоб він перенаправляв клієнтів на підконтрольний зловмиснику сервер. У результаті користувач, намагаючись отримати доступ до легітимного ресурсу, потрапляє на фальшиву копію, яка може збирати облікові дані, поширювати шкідливе ПЗ або використовуватися для фішингу. Алгоритм

атаки включає підготовку підроблених DNS-відповідей і їх масове надсилання у той момент, коли сервер виконує резолвінг. Інструменти на кшталт dnsspoof або DNSChef допомагають автоматизувати цей процес. Захист передбачає впровадження DNSSEC, використання лише довірених рекурсивних DNS-серверів, моніторинг аномальних змін DNS-записів, а також ізоляцію внутрішніх DNS від публічного доступу.

Уразливості маршрутизаторів

Маршрутизатори та інші мережеві пристрої є особливо привабливими для атак, оскільки контроль над ними дозволяє зловмиснику здійснювати повний моніторинг або перенаправлення трафіку. Вразливості часто пов'язані з бекдорами у прошивках, незахищеними веб-інтерфейсами керування, дефолтними паролями чи відсутністю оновлень від виробника. Типовий алгоритм атаки починається зі сканування інтернет-простору (Masscan, Shodan, Zoomeye) для виявлення відкритих інтерфейсів маршрутизаторів. Далі перевіряються відомі CVE для конкретної моделі, і за допомогою RouterSploit чи Metasploit виконується експлуатація вразливості. У разі успіху зловмисник може змінити конфігурацію пристрою, налаштувати переадресацію DNS-запитів або створити бекдор для віддаленого доступу. Захист від таких атак включає регулярне оновлення прошивок, вимкнення публічного доступу до інтерфейсів адміністрування, застосування VPN або Zero Trust для віддаленого керування, сегментацію мережевого трафіку та моніторинг конфігураційних змін.

Таким чином, програмні вразливості охоплюють широкий спектр загроз — від класичних переповнень буфера до складних криптографічних та мережевих помилок. Їхня експлуатація завжди базується на чітких алгоритмах дій зловмисників і наборі відпрацьованих інструментів, а ефективний захист вимагає багаторівневого підходу: своєчасного патч-менеджменту, превентивних технологій, захищених конфігурацій і постійного моніторингу.

Експлойти та їх класифікація

Експлойт — це спеціально створений код або послідовність дій, що використовує вразливість для досягнення несанкціонованого результату: від виконання довільного коду до отримання привілеїв адміністратора. Відомі категорії експлойтів: локальні (використовуються на вже скомпрометованому хості), віддалені (Remote Code Execution), DoS-експлойти для виведення сервісів з ладу, а також експлойти на основі веб-уразливостей (SQLi, XSS).

Експлойти — це спеціально підготовлені програми або послідовності дій, які використовують конкретні вразливості для досягнення несанкціонованого результату, і їхнє розуміння вимагає одночасного уявлення про техніку атаки, характерні індикатори і набір захисних контрзаходів. Класифікація експлойтів відображає вектор і очікуваний результат атаки; кожен клас має свої особливості алгоритму дій, типовий інструментарій, характерні ознаки в мережі та на хості й набір ефективних заходів з захисту та виявлення.

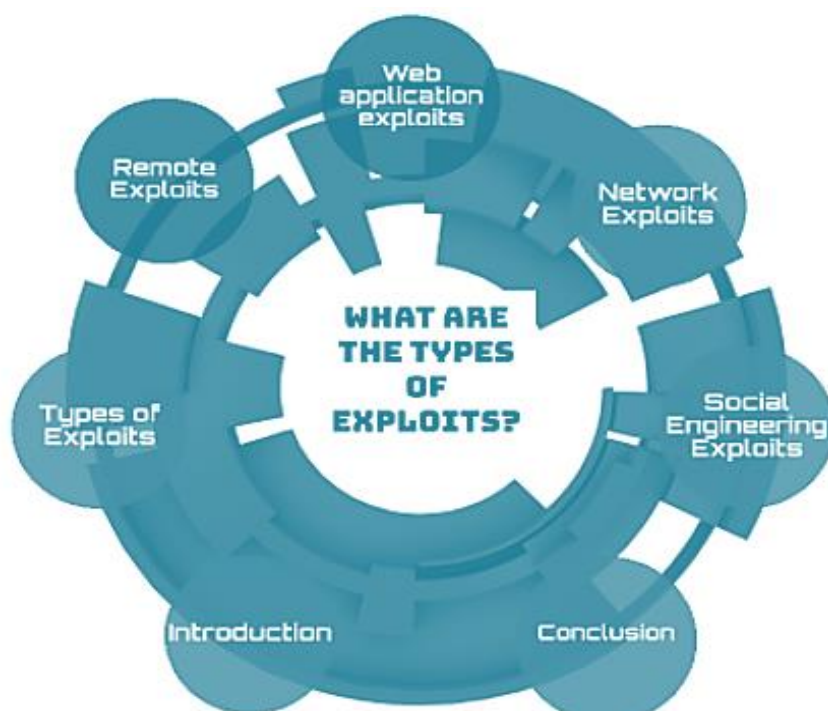


Рис. 3.24. Кругова класифікація типів експлойтів
Джерело: [1-9, 11-26, 32-48, 60-65, 72-77]

Рисунок 3.24 подано у вигляді тривимірного колеса: у центрі — основна теза про експлойти як категорію атак, а навколо неї розташовані сегменти і вкладені кружечки з підпунктами — Web application exploits, Network exploits, Social engineering exploits, Remote exploits та загальний маркер Types of exploits; додаткові мітки Introduction і Conclusion позначають логічний обрамляючий вступ і висновок для презентації. Композиція підкреслює, що категорії експлойтів не ізольовані — вони перетинаються і можуть комбінуватися в ланцюги атак (наприклад, веб-вразливість → віддалений RCE → локальне підвищення привілеїв), тому колесо зручно читати як набір взаємопов'язаних векторів загрози. Графічно сегменти оформлені контрастно й дають змогу для кожної області додати короткі підписи: типові вектори доставки, характерні індикатори (мережеві

й хост-сигнали) і базові контрзаходи (патчинг, WAF, EDR, навчання користувачів тощо). Така діаграма добре підходить для слайду-огляду: вона відразу дає аудиторії структуру теми, а кожен сегмент легко розгортається в окрему підсекцію із прикладами (SQLi/XSS для веба, DDoS для мережі, фішинг для соціальної інженерії тощо). В презентації можна додати стрілки або анотації, що ілюструють можливі «чейнінги» між сегментами, а також позначити пріоритети захисту для конкретного середовища.

Експлойти віддаленого виконання коду (Remote Code Execution, RCE)

Експлойти віддаленого виконання коду (Remote Code Execution, RCE) спрямовані на те, щоб через мережевий інтерфейс або інтерпретатор примусити віддалену систему виконати довільні інструкції атакуючого. Типовий алгоритм включає розвідку (виявлення служби та її версії), підбір відповідного експлойта або PoC, відправку спеціально сформованого мережевого запиту чи пакета, який активує вразливу обробку даних у серверній програмі, і доставку payload — шелу, бекдора або другого етапу розгортання. Інструменти, які використовують як дослідники та зловмисники, — Metasploit, готові модулі з Exploit-DB, скрипти PoC на Python чи Ruby; для веб-векторів — Burp Suite у поєднанні зі спеціалізованими плагінами. Індикаторами RCE можуть бути аномальні HTTP/SMB/SMTP-запити з нетиповими полями або довгими параметрами, запуск незвичних процесів після обробки зовнішнього запиту, підозріла мережева активність з хоста, що раніше її не мав. Захист проти RCE спирається на багатослойність: регулярний патчинг і відключення вразливих сервісів, застосування WAF зі строгими правилами, використання runtime-захисту (sandboxing контейнерів, AppArmor/SELinux), обмеження привілеїв сервісних облікових записів та EDR для виявлення підозрілих запусків процесів; додатково корисні технології DEP/ASLR/Control Flow Integrity у бінарях для ускладнення експлуатації пам'яті.

Локальні експлойти (local exploits)

Локальні експлойти (local exploits) орієнтовані на систему, яку атакувальник вже дещо скомпрометував (наприклад, має користувацький обліковий запис) і прагне підвищити привілеї або закріпитися в системі. Алгоритм типово починається з аналізу локального оточення (права, setuid-бінарники, незахищені служби), пошуку misconfiguration або уразливих модулів ядра, застосування підготовленого експлойта для підйому привілеїв і нарощування постійності. Інструменти, що використовуються, включають локальні PoC, набори для підйому привілеїв у Linux/Windows і утиліти на кшталт Mimikatz для вилучення облікових даних. Ознаками використання

локального експлойта є раптове підвищення прав процесів, поява нових служб чи акаунтів, незвична активність у системних журналах. Захист від локальних експлойтів ґрунтується на принципі мінімальних прав, обмеженні і моніторингу виконання setuid/адмінських програм, захисті LSASS та інших чутливих об'єктів, використанні апаратного та ОС-рівневого hardening (secure boot, kernel patch protection) і суворому контролі змін у системі.

DoS-експлойти

DoS-експлойти спрямовані не на отримання контролю, а на виведення сервісу з ладу або істотне погіршення його працездатності. Алгоритм DoS може бути простим (відправка великого потоку запитів) або складним (надсилання спеціалізованих malformed-пакетів, що викликають витік пам'яті чи деградацію). Інструменти варіюються від простих load-генераторів (LOIC/HOIC у публічних сценаріях) до складних ботнетів і спеціалізованих скриптів, що використовують уразливості стеку протоколів. Симптоми DoS-атаки — різке зростання мережевого трафіку, перевантаження CPU/пам'яті сервера, падіння кількості обслуговуваних сесій або регулярні краші процесів. Для захисту застосовують архітектурні контрзаходи: масштабування і балансування навантаження, rate-limiting, CDN/edge-фільтрацію, спеціалізовані DDoS-сервіси постачальників, а також IDS/IPS правила, що виявляють аномальні шаблони трафіку.

Експлойти веб-орієнтованого типу (SQL Injection, XSS, CSRF, Command Injection тощо)

Експлойти веб-орієнтованого типу (SQL Injection, XSS, CSRF, Command Injection тощо) базуються на недоліках у валідації вхідних даних або у некоректній обробці параметрів додатком. Алгоритм для SQLi передбачає аналіз поведінки параметрів запитів, інжекцію конструкцій, які змінюють або виводять дані з бази, і отримання конфіденційної інформації або ескалацію доступу. Для XSS — пошук полів, де відображений HTML/JS не проходить належної санації, і доставка payload, що виконається в браузері жертви (крадіжка сесій, фішинг). Інструменти для виявлення й експлуатації веб-вразливостей — Burp Suite, OWASP ZAP, sqlmap, різні браузерні скрипти і плагіни. Показовими індикаторами є незвичні параметри у HTTP-запитах, дивні помилки SQL чи багатократні запити з повторним інжектом. Захист від веб-експлойтів базується на принципах безпечного кодування: застосуванні підготовлених SQL-запитів і ORM, належній санітизації та ескейпінгу вхідних даних, впровадженні Content Security Policy (CSP), використанні WAF з правилом для OWASP Top10, регулярному скануванню DAST/SAST і захищеному циклі розробки (secure SDLC).

Існують також експлойти клієнтської сторони

Існують також експлойти клієнтської сторони, які атакують браузер, плагіни або офісні пакети: їхній алгоритм починається зі створення приманки (фішингові листи, сайти) і завершення доставкою exploit-chain, що використовує баги у парсерах документів або рушіях JavaScript для отримання RCE на машині користувача. Захист у цьому випадку — оновлення клієнтів і плагінів, застосування ізоляції браузера (sandbox, browser isolation), блокування макросів та заготовок документів, EDR на кінцевих точках і користувацька обізнаність.

«Чейнінг»-експлойти

Окрему категорію становлять «чейнінг»-експлойти, коли декілька вразливостей комбінуються у ланцюг (наприклад, веб-RCE → Mimikatz → AD-pivot), щоб досягти цілі; алгоритм тут передбачає ретельне планування, збір контекстної інформації та поетапне розгортання інструментів для розширення контролю. Інструменти для таких складних кампаній включають фреймворки C2 (Cobalt Strike), скрипти для автоматизації, модулі для ескалації привілеїв і засоби для персистенції. Виявлення складних ланцюгів вимагає кореляції подій у SIEM/EDR/NDR, аналізу графів взаємодій (graph analytics) та використання TIP для зв'язування елементів кампанії.

Експлойт-кити та «комерційні» набори

Експлойт-кити та «комерційні» набори значною мірою автоматизують етапи підбору та реалізації атак, вони часто розповсюджуються у чорних ринках і надають менш кваліфікованим нападникам можливість масштабувати діяльність. Хоча деталі роботи таких наборів залежать від конкретного продукту, загальна закономірність — автоматичне виявлення вразливих сервісів, підбір експлоїтів і доставка payload з C2-каналом. Захист від exploit-kit-кампаній — це поєднання веб-захисту (WAF, CDN), оновлень клієнтів і серверів, блокування підозрілих доменів і аналітики мережевого поведінкового трафіку.

Захист від експлоїтів

Захист від експлоїтів передбачає впровадження систем управління вразливостями (Vulnerability Management): регулярні сканування мережі (Nessus, OpenVAS), перевірка на відповідність базам CVE, застосування патчів та оновлень, сегментація критичних ресурсів. Важливу роль відіграє також принцип мінімальних привілеїв, багатофакторна автентифікація та використання IDS/IPS і WAF для блокування спроб експлуатації.

Таким чином, вразливості є невід'ємною частиною життєвого циклу будь-якої інформаційної системи, а експлойти — головний інструмент атакуювальників. Комплексний підхід, що включає своєчасне оновлення,

моніторинг і проактивне виявлення Zero-Day атак, дозволяє мінімізувати ризики та підвищити кіберстійкість комп'ютерних мереж.

Захист від експлойтів має бути багаторівневим і системним, адже лише поєднання технічних, процесних і організаційних заходів робить експлойт значно менш ефективним. Першим і фундаментальним шаром є система управління вразливостями (Vulnerability Management), яка забезпечує безперервне виявлення, оцінку та усунення відомих слабких місць: регулярні внутрішні й зовнішні сканування (Nessus, OpenVAS та інші), кореляція результатів зі списками CVE/NVD, пріоритизація виправлень за CVSS у поєднанні з бізнес-контекстом і розгортання патчів з контрольованим roll-out. Важливо не тільки запускати сканер, а й мати автоматизований pipeline — інтеграцію сканера з CMDB/asset-inventory, ticketing (ServiceNow/Jira) і процесами change-management, щоб у кожного критичного виявленого CVE був визначений власник, SLA на виправлення і план відкату. Метрики, які при цьому відслідковуються, — «vulnerability age», Mean Time To Patch (MTTP) за критичністю і частка систем, що відповідають базовим стандартам безпеки.

Другий шар — hardening і превентивні контрзаходи на рівні операційної системи та застосунків. Тут застосовують secure-by-default конфігурації (CIS Benchmarks, STIG), обмежують набори запущених сервісів, видаляють дефолтні облікові записи, впроваджують принцип least-privilege для сервісних акаунтів і використовують РАМ-рішення для управління привілеями та ревокації доступу. На рівні бінарів і рантайму необхідно впроваджувати техніки, які ускладнюють експлуатацію пам'яті: ASLR, DEP/NX, stack canaries, Control Flow Integrity та інші механізми mitigations, а також компіляцію з флагами безпечних лінкерів і вимкнення небезпечних функцій. Ізоляція сервісів через контейнеризацію чи віртуалізацію і застосування політик AppArmor/SELinux додає ще один бар'єр — навіть якщо процес буде експлуатовано, його можливості щодо шкоди будуть обмежені.

Третій шар — мережеві архітектури та сегментація. Сегментація (VLAN, VRF) і мікросегментація в хмарі з чіткими ACL/NSG значно ускладнюють латеральний рух після успішної експлуатації. Архітектурні патерни zero-trust, де доступ не надається «за замовчуванням», а кожен потік автентифікується і авторизується, зменшують ризик, що один скомпрометований сервіс приведе до повного доменного компромісу. На периметрі та в точках входу варто застосовувати WAF і edge-фільтрацію для блокування відомих векторів веб-експлуатації, а також виконувати TLS-inspection там, де це дозволяє політика — щоб виявляти шкідливі пейлоади у зашифрованих сесіях. Patching на мережевих пристроях та вразливих

IoT/роутерів, обмеження доступу до пристроїв адміністрування тільки через secure bastion/jump-hosts і VPN/Zero Trust доступ — це додаткові критичні елементи.

Четвертий шар — виявлення та реагування. Оскільки сигнатури не завжди покривають нові експлойти, необхідно фокусуватися на поведінкових індикаторах: EDR на кінцевих точках для детектування нетипових системних викликів, ін'єкцій пам'яті чи підозрілих дочірніх процесів; NDR для моніторингу аномалій мережевого трафіку, DGA-подібних DNS-запитів або регулярних «маячків» до незнайомих доменів; SIEM для кореляції подій з різних джерел і UEBA для виявлення атипової поведінки користувачів і ентиті. Важливим елементом є повернення наданої телеметрії у відкритий аналіз: збереження memory dumps, PCAP, журналів процесів і командних рядків з достатньою ретеншеном, щоб забезпечити глибоку форензичну аналітику при підозрі на експлуатацію.

П'ятий шар — процеси реагування та автоматизація. Потрібні заздалегідь підготовлені runbook-и і playbook-и в SOAR, які автоматично виконують containment-дії: ізоляцію хоста, вилучення мережевого інтерфейсу, блокування конкретних доменів чи IP у фаєрволі, створення тикета на Patch-team. Автоматизація зменшує Mean Time To Contain і дозволяє швидко застосовувати virtual-patches (WAF-rules, ACL-обмеження) до того, як виробник випустить офіційний патч. Водночас важливо мати процеси ескалації: контакти вендора, CERT-повідомлення, і політику disclosure/coordination, що дозволяє працювати з вендором над виходом патча.

Шостий напрямок — безпечна розробка і тестування. Secure SDLC з інтегрованими SAST/DAST, dependency-scanning (SBOM), fuzz-тестуванням і threat modeling зменшують появу нових вразливостей. CI/CD пайплайни мають включати gating: не дозволяти релізи з критичними вразливостями, автоматично запускати тестування статичного й динамічного аналізу та перевірку third-party бібліотек. Для open source компонентів слід вести bill-of-materials і контролювати оновлення залежностей.

Сьомий — підготовка кадрового ресурсу і тестування готовності. Регулярні pentest-и, red/purple-team вправи і BAS (Attack Simulation) дозволяють виявити слабкі місця в конфігураціях і в логіці реагування, тестувати runbook-и та оцінювати охоплення детекцій. Навчання розробників secure-coding практикам, операторів SOC і команд інцидент-респонсу підвищує загальну здатність організації протистояти експлойтам.

Нарешті, не менш важливою є інтеграція Threat Intelligence: підписка на фіди, автоматичне збагачення подій TIR-даними і аналіз приватних та

відкритих розкриттів дозволяють оперативно виявляти сигнали про нові експлойти й Zero-Day кампанії. Це поєднується з політиками контрактів постачальників (вимоги щодо безпеки, SLA на патчі) і практиками disclosure/bug-bounty для прискорення виявлення вразливостей у ланцюжку постачання.

В підсумку, ефективний захист від експлойтів — це не одна технологія, а комплекс заходів: від інвентаризації й оперативного патчингу до рантайм-мітгейшн і поведінкового моніторингу, від secure SDLC до відпрацьованих playbook-ів і координації з вендорами. Лише такий системний, циклічний підхід з метриками (MTTD, MTTR, MTTP, % patched) і постійним тестуванням забезпечить суттєве зменшення ризику успішних експлойт-атак у складних корпоративних інфраструктурах.

Zero-Day атаки

Окрему категорію становлять Zero-Day вразливості, тобто ті, що ще не були публічно розкриті й не мають виправлення від виробника. Такі атаки є найнебезпечнішими, оскільки організації не мають готових сигнатур для IDS/IPS чи оновлень безпеки. Вони активно використовуються АРТ-групами для шпигунства, саботажу або кібервійни. Zero-Day можуть поширюватися через фішинг, шкідливі документи або мережеві експлойти, що вражають популярні сервіси.

Зображений рисунок відображає послідовність етапів, через які проходить Zero-Day атака: від моменту виявлення нової вразливості до її виправлення виробником. На початковому етапі зловмисники знаходять невідому помилку у програмному забезпеченні, яка ще не задокументована вендором і не має патча. Далі формується експлойт, що використовує цю уразливість, після чого здійснюється його доставка до цільової системи — через фішингові повідомлення, заражені файли, мережеві експлойти або ланцюги постачання. У фазі експлуатації відбувається виконання шкідливого коду, закріплення в системі, розширення доступу та збір або ексфільтрація даних. Наступним етапом є активне використання вразливості, яке може тривати від кількох днів до місяців, доки вендор не отримає інформацію про проблему й не підготує оновлення. Завершальною фазою життєвого циклу є випуск патча та поширення його серед користувачів, проте навіть після цього атака може зберігати ефективність проти тих систем, які не встигли оновитися. Такий життєвий цикл показує, чому Zero-Day є настільки небезпечними: організації фактично залишаються без готових механізмів захисту у проміжку між відкриттям вразливості та випуском патча.

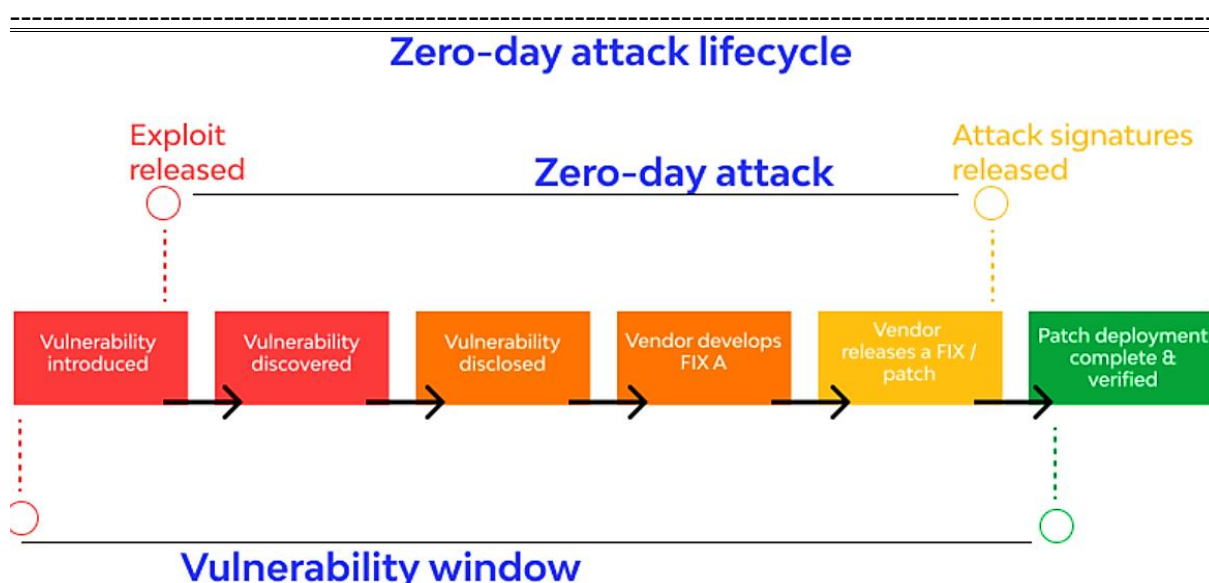


Рис. 3.25. Багаторівнева архітектура SDN (Software-Defined Networking)
Джерело: [1-9, 11-26, 32-48, 60-65, 72-77]

Zero-Day атаки — це особлива й дуже небезпечна категорія нападів, бо вони спираються на вразливості, які ще не були оприлюднені й не мають офіційного виправлення від вендора; через відсутність патча й сигнатур такі атаки часто проходять повз традиційні сигнатурні засоби захисту і застосовуються для цілеспрямованих кампаній шпигунства, саботажу або кібервійни. Принцип дії Zero-Day полягає в тому, що атакувальник знаходить або купує вразливість у програмі/бібліотеці/прошивці, розробляє експлойт, ретельно його тестує в умовах, близьких до цільового середовища, а потім доставляє payload жертві через найбільш ефективний вектор — цілеспрямовані фішингові емейли зі шкідливими документами, водопровідні ланцюги постачання (supply chain), безпосередні мережеві експлойти на інтернет-маршрутизатори чи публічні сервіси або через заражені сторонні оновлення. Типовий алгоритм кампанії Zero-Day має вигляд послідовності: збір цільової розвідки для визначення софту й конфігурацій, підбір вектора доставки з урахуванням бізнес-процесів жертви, тонка адаптація експлойта під ціль (з урахуванням версій ПЗ і захисних механізмів), доставлення атаки, пост-експлуатаційні дії (закріплення доступу, ескалація привілеїв, латеральний рух, ексфільтрація) і активне приховання слідів. Часто Zero-Day експлуатуються у вигляді «чейнінгу» — комбінації кількох вразливостей (наприклад, веб-RCE → локальний ескалатор → credential dumping → AD pivot), щоб домогтися стійкого доступу й виконати цілі кампанії.

Інструментарій для реалізації Zero-Day атак

Інструментарій для реалізації Zero-Day кампаній відрізняється від «масових» наборів тим, що велику роль відіграє кастомізація й утиліти для бекдору та C2-зв'язку: на етапі розробки та тестування застосовують приватні стенди і дебагери, fuzz-інструменти для пошуку пам'ятевих дефектів, реверсінг-набори для аналізу бінарів і мережні сканери для точного визначення таргетних конфігурацій. Для доставки й подальшого управління атакою використовують спеціалізовані C2-фреймворки або приватні тулкити, інструменти для створення «легітимного» трафіку (щоб маскувати маячки), обфускації коду й стеганографії в мережі. На ринку існує також чорний сегмент, де Zero-Day експлойти купуються/продаються через брокерів або «експлойт-маркетплейси», що дозволяє навіть невеликим угрупованням отримати доступ до високоякісних експлойтів.

Індикатори компрометації при Zero-Day

Індикатори компрометації при Zero-Day дуже рідко схожі на класичні сигнатури; тому SOC іде не шляхом пошуку конкретного байткоду, а фокусується на поведінкових аномаліях. До ознак, що можуть вказувати на використання Zero-Day, належать раптові та незрозумілі збільшення кількості падінь процесів або сервісів, аномальні звернення до внутрішніх API із незвичними довгими параметрами, поява нових та невідомих бінарників/модулів, нетипові операції з пам'яттю (великі одноразові алокації чи інтенсивні операції читання/запису), довготривалі або шифровані сесії до невідомих зовнішніх доменів і незвичні послідовності системних викликів. Також варто звертати увагу на непрямі індикатори: нетипові поведінкові зміни користувачів або сервісів (наприклад, раптове витягування великих обсягів даних у незвичний час), кореляція подій між кінцевою точкою й мережею, і сигнали з Threat Intelligence про нові, ще непатчені вектори, що циркулюють у приватних колах.

Захист від Zero-Day атак

Захист від Zero-Day атак потребує багатовекторного й проактивного підходу. По-перше, архітектура повинна мінімізувати «поверхню атаки»: сегментація мережі й застосування принципу least-privilege не дозволять експлойту швидко розширити контроль за мережею. По-друге, впровадження компесуючих контролів («virtual patching») — наприклад, правила WAF, фільтрація на крайових пристроях чи тимчасові мережеві ACL — дозволяють блокувати характерні вектори доставки до того, як з'явиться офіційний патч. По-третє, потрібно розгорнути сучасні засоби моніторингу з фокусом на поведінку: EDR і NDR з можливістю виявлення аномалій у виконанні процесів і мережевих шаблонах, SIEM-кореляції, UEBA-моделі для виявлення нетипових дій користувачів і ентиті. По-четверте, важливими є техніки запобігання експлуатації пам'яті і

ускладнення експлойтів: DEP/ASLR, Control Flow Integrity, stack canaries, ізоляція сервісів у контейнерах чи віртуальних машинах, використання sandbox для обробки потенційно небезпечних артефактів (приклад — відкриття документів у захищеному середовищі). По-п'яте, необхідна оперативна інтеграція з Threat Intelligence: раннє отримання сигналів про появу приватних експлойтів чи згадки в темних каналах може дати перевагу і дозволить негайно розгорнути детект- та мітгейшн-заходи. По-шосте, практики purple-/red-team та регулярне fuzzing-тестування важливі для виявлення власних «zero-day-подібних» слабких місць до того, як ними скористається ворог.

У випадку виявлення потенційного Zero-Day інциденту правильна процедура реагування — критична. Перші кроки включають негайну ізоляцію підозрілих хостів, створення пам'яті-форензики (memory dump) та захоплення мережових PCAP-записів для подальшого аналізу, збагачення знахідок через TIR і координацію з вендором для отримання патча або рекомендацій. Тимчасові заходи можуть включати застосування virtual patch, блокування пов'язаних доменів/адрес, обмеження функціоналу у вразливих сервісах і розгортання детекторів-наглядачів для специфічних поведінкових патернів, виявлених при інциденті. Важливою є також політика відповідального повідомлення: співпраця з виробником і, за потреби, з CERT/ISAC спільнотами для координації виправлення.

Організаційно-процесні заходи для зниження ризику Zero-Day включають впровадження програми управління вразливостями з пріоритизацією за CVSS і бізнес-контекстом, контрактні вимоги до постачальників щодо безпеки та швидкого патчингу, участь у bug-bounty програмах і стимулювання зовнішнього пошуку вразливостей, а також регулярні навчання для SOC і інженерів на тему розпізнавання нетипових індикаторів. Для критичних систем доцільно застосовувати defense-in-depth, включаючи апаратні засоби захисту (TPM, secure boot), сегментацію, ретельну валідацію поставок коду (software bill of materials, SBOM) і перевірку цифрових підписів оновлень.

Підсумовуючи, Zero-Day атаки мають високу потенційну шкоду через відсутність готових патчів і сигнатур, тому основний захист проти них — це архітектурна стійкість, проактивний моніторинг поведінки, оперативна співпраця з вендорами і TIR, можливість швидко застосувати virtual patching та продемонструвати готові runbook-процедури для швидкого реагування і форензики; тільки комбінація технічних, організаційних і процесних заходів дає шанс мінімізувати вплив Zero-Day на критичну інфраструктуру.

3.8. ІНСТРУМЕНТИ МОНІТОРИНГУ Й ВИЯВЛЕННЯ АТАК: IDS/IPS, SIEM, EDR, NDR

Інструменти моніторингу й виявлення атак — IDS/IPS, SIEM, EDR і NDR — складають ядро сучасної моделі захисту, забезпечуючи збір телеметрії, раннє виявлення, кореляцію подій і першу реакцію на інциденти. Вони доповнюють одне одного: SIEM централізує та корелює логи, EDR дає глибоку видимість кінцевих точок, NDR фіксує мережеві аномалії й потоки, а IDS/IPS (вбудовані або інлайн) забезпечують швидку блокуючу або детектуючу дію на рівні мережі. Нижче наведено детальний, взаємопов'язаний опис кожного класу систем, їхніх принципів роботи, архітектури розгортання, типів детекцій, джерел телеметрії, практичних рекомендацій щодо налаштування й інтеграції та обмежень, про які слід пам'ятати.

IDS/IPS Системи виявлення і запобігання вторгнень (IDS — passive, IPS — inline/active)

Системи виявлення і запобігання вторгнень (IDS — passive, IPS — inline/active) працюють переважно на мережевому рівні і аналізують пакети, сесії та протокольні шаблони щоб ідентифікувати відомі атаки або підозрілі шаблони трафіку. Принцип дії оперує двома основними підходами: сигнатурний аналіз, який порівнює мережеві пакети й потокові метадані з відомими підписами (наприклад, Suricata/Snort правила), і евристичний/аналітичний аналіз, який шукає аномалії у поведінці протоколів або нетипові послідовності. IDS зазвичай розгортається пасивно (span/mirror порт або TAP) для моніторингу, а IPS — inline для миттєвого блокування або відкидання трафіку. Ключові джерела даних — full packet capture (PCAP), розпізнавані протоколи (HTTP, DNS, SMB), TLS/SSL хендшейки та NetFlow/flow-маркерні дані. Основні індикатори, які IDS/IPS виявляють: підозрілі payload'и в HTTP, відомі експлойт-патерни, DNS-запити із високою ентропією (можливий DGA), ARP-spoofing чи аномальні TTL/фрагментації. При розгортанні важливо продумати placement (периметр, DMZ, віртуальні TAP у хмарі), capacity planning (пропускна здатність inline), регулярне оновлення сигнатур і механізми виключень (whitelisting) щоб зменшити фальшпозитиви. IDS/IPS добре інтегруються з SIEM та NDR: події з мережевого сенсора збагачуються контекстом хостів і користувачів, що полегшує пріоритизацію.

IDS/IPS також мають свої обмеження й виклики у використанні. Сигнатурний підхід не здатний виявляти zero-day та поліморфні атаки, оскільки вони ще не мають відомих патернів. Евристичний аналіз, у свою чергу, часто породжує велику кількість фальшивих спрацювань, що вимагає

ручної перевірки та грамотного fine-tuning правил. Тому оптимальна ефективність досягається лише у поєднанні IDS/IPS із іншими шарами безпеки: EDR для аналізу поведінки на кінцевих точках, SIEM для централізованої кореляції подій та NDR для глибшого аналізу потоків. Сучасні IPS підтримують інтеграцію з threat intelligence feeds для оперативного отримання нових сигнатур і IoC, а також мають API для автоматизації реакцій через SOAR (наприклад, блокування IP або відключення користувача в Active Directory). Захист додатково підсилюється технологіями TLS-inspection, які дають змогу IDS/IPS бачити вміст зашифрованих потоків, хоча це створює баланс між безпекою та приватністю. Практичне впровадження IDS/IPS передбачає також побудову baseline нормальної мережевої активності: це допомагає SOC-команді швидше ідентифікувати відхилення, пов'язані з атаками. Важливим елементом залишається й тестування системи шляхом регулярних pentest і red-team вправ, що дають змогу перевірити, чи дійсно сенсори спрацьовують на актуальні техніки з MITRE ATT&CK. Таким чином, IDS/IPS не є ізольованим інструментом, а ефективним компонентом у стратегії defense-in-depth, який у поєднанні з іншими системами формує цілісний підхід до виявлення та запобігання сучасним кіберзагрозам.

SIEM

Система централізованого збору, нормалізації, кореляції й збереження логів — серце SOC-операцій. SIEM приймає логі телеметрії з EDR, NDR, IDS/IPS, проксі, VPN, хмарних сервісів, IAM, додатків і формує корельовані інциденти, дашборди й тригери на основі правил або аналітики. Принцип дії SIEM складається з побудови конвеєра прийому (collect → parse → normalize), збагачення (threat intel, asset tags, owner), кореляції (rule-based або поведінкова кореляція) і збереження з ретеншеном для розслідувань. SIEM дозволяє реалізувати складні детекційні сценарії: наприклад, поєднати численні невдалі логіни (IAM) + підозріле викликання PowerShell (EDR) + нетиповий вихідний трафік (NDR) → створити high-severity alert. Типи детекцій у SIEM: сигнатурні кореляції (Sigma-подібні правила), тимчасові аномалії (time-window baselines), загрози на основі інтелекту (IOC match) і ML/UEBA-аналітика. Практичні аспекти: нормалізація логів (спільна таксономія подій), чіткі правила пріоритизації щоб уникнути overload, адаптація retention-політик до регуляторних вимог, інтеграція з TIP для автоматичного збагачення IoC та з SOAR для автоматизованих playbook-ів. Обмеження SIEM — залежність від якості телеметрії і правилами, які потребують постійного тюнінгу; відсутність EDR/NDR-типової глибини без додаткових джерел.

SIEM є не лише інструментом для збору логів, а й центральним елементом у процесі реагування на інциденти та форензики. У практичному використанні він забезпечує створення єдиного джерела правди (single source of truth) для всієї організації: усі журнали подій та безпекові індикатори зберігаються централізовано й доступні для ретроспективного аналізу. Це особливо критично при розслідуванні складних інцидентів, коли потрібно відновити ланцюг подій за тижні чи місяці. Сучасні SIEM-платформи (Splunk, IBM QRadar, Azure Sentinel, Elastic Security) підтримують інтеграцію з UEBA-модулями для поведінкового аналізу користувачів і машин, а також застосування машинного навчання для виявлення складних і прихованих атак. Додатково вони дозволяють будувати case management: групувати пов'язані алерти, призначати їх конкретним аналітикам SOC і відслідковувати статус обробки. Важливим аспектом є масштабованість і продуктивність: високонавантажені підприємства генерують терабайти логів щодня, тому SIEM має бути оптимізований під швидкий пошук, індексацію та архівування. Разом із цим виникають виклики — від вартості ліцензування за обсяг даних до необхідності постійного fine-tuning правил кореляції, без чого система або «тоне» у фальшивих тривогах, або пропускає реальні атаки. Отже, SIEM виступає не ізольованим продуктом, а платформою інтеграції: його сила розкривається у зв'язці з EDR, NDR, IDS/IPS та SOAR, що перетворює масивну телеметрію на релевантні інциденти й пришвидшує реагування SOC-команди.

EDR (Endpoint Detection and Response)

Endpoint Detection and Response фокусується на кінцевих точках: процесах, виконаннях, реєстрі, мережевих викликах процесів, ін'єкціях у пам'ять та активності командних рядків. EDR збирає детальну телеметрію з хостів і дозволяє не тільки виявляти підозрілу поведінку (execution anomalies, persistence creation, credential dumping), але й оперативно реагувати (ізоляція хоста, завершення процесів, збір memory dump). Принцип дії EDR — агентна телеметрія, локальна або хмарна аналітика, індикатори IOC-хешів і поведінкові моделі. Методи детекції включають сигнатури (YARA-подібні правила), поведінкові детектори (послідовності системних викликів, підозрілі child-process chains), машинне навчання для виділення аномальних профілів процесів і зв'язки з threat intelligence. Приклади індикаторів: запуски PowerShell із EncodedCommand, інжекції в LSASS, незвичні DLL loads, дивні persistence entries у автозавантаженнях. Розгортання EDR вимагає охоплення всіх критичних хостів, керування політиками (whitelist/blacklist), чіткої інтеграції з SIEM/SOAR та процедур реагування для автоматизації containment. EDR критично залежить від

агента — слід стежити за його сумісністю, впливом на продуктивність і стійкістю до атак (self-protection).

EDR — це не лише сенсор на хості, а повноцінна платформа операцій безпеки, тож її ефективність визначається якістю покриття, операційною зрілістю та гігієною кінцевих точок. На практиці слід поєднувати EDR із базовим EPP/антивірусом і політиками hardening (Application Control/AppLocker, PowerShell Constrained Language Mode, контроль макросів, блокування LOLBins), визначити обов'язкове охоплення всіх серверів, робочих станцій і «складних» середовищ (macOS/Linux, VDI, контейнерні та хмарні ворклоади), а також упровадити tamper protection і self-protection агента, щоб ускладнити його вимкнення під час атаки. Важливо налаштувати збирання достатньої, але не надмірної телеметрії: події процесів, модулів, мережевих з'єднань, реєстру/автозапусків, драйверів і скрипт-інтерпретаторів із ретеншеном, який відповідає вимогам розслідувань; окремо передбачити офлайн-буферизацію для ноутбуків поза корпоративною мережею. Операційно EDR має бути «вплетений» у playbook-и: автоматична ізоляція мережі хоста за високою впевненістю, блокування процесів і хешів, збирання live-response артефактів (memory dump, triage), а далі — ескалація в SOAR і SIEM для кореляції з NDR/IDS подіями. Для проактивності варто використовувати вбудовані мови запитів (KQL/SPL-подібні) і створювати hunt-запити за TTP MITRE ATT&CK (аномальні child-process chains для Office/Browser, підозрілі AMSti/ETW обходи, створення нестандартних служб, PsExec/WinRM-шаблони), регулярно тюнити політики за результатами purple-team вправ. Серед обмежень — залежність від агента та прав на хості, можливі прогалини видимості на рівні ядра/драйверів, ризики приватності (PPI у командах і шляхах файлів), а також вплив на продуктивність при агресивних політиках; їх зменшують профілювання навантаження, поетапні розгортання, винятки з підписами й чіткий RACI для власників активів. Для 24×7 покриття доцільно додати MDR-сервіс або власний черговий L1/L2, а показники ефективності тримати у фокусі: MTDD/MTTR, % охоплення агентом, частка автоматизованих containment-дій, частка фальшпозитивів та час відкату/оновлення агента на критичних системах.

NDR (Network Detection and Response)

Network Detection and Response концентрується на потоковій та повно-пакетній мережевій телеметрії (NetFlow/IPFIX, PCAP, TLS fingerprint, HTTP headers, DNS logs) і використовує евристичні, статистику та ML для виявлення аномалій, які можуть вказувати на компрометацію або C2-комунікацію. NDR-компоненти часто застосовують поведінкові моделі (baseline flows per host, entropy analysis DNS, beaconing detection) і graph-

аналіз для виявлення lateral movement та складних ланцюгів атак. Принцип дії NDR передбачає збір потоків/пакетів, побудову сесій, збагачення (asset tagging, user mapping), виявлення патернів і формування детектів з подальшою інтеграцією в SIEM/EDR. Специфічні детекції NDR: DGA/DNS anomalies, long-lived encrypted sessions без взаємодії користувача (підозра на тунелювання), підвищена ентропія DNS запитів (функції DGA), lateral movement fingerprint (SMB/LDAP patterns), SMB relay attempts, та C2 beacon detection (регулярні “маячки” з визначеними інтервалами). Розгортання NDR вимагає mirror/span портів або TAP-ів, коректного asset mapping між мережевими IP і хост-ідентифікаторами (для effective triage), а також уваги до приватності (TLS inspection має юридичні й етичні обмеження). NDR сильний у виявленні тих технік, які обминають сигнатурні системи EDR (living-off-the-land, fileless C2).

Попри свої переваги, NDR має і певні виклики: великий обсяг мережових даних вимагає потужної інфраструктури для зберігання та аналітики, а також кваліфікованих аналітиків SOC для правильного тлумачення результатів. Ще однією складністю є широке застосування шифрування (TLS 1.3, DoH/DoT), що обмежує можливості глибокого аналізу пакетів і змушує покладатися на метадані або TLS fingerprinting. Для збереження ефективності NDR необхідно регулярно оновлювати моделі машинного навчання, коригувати базові профілі поведінки користувачів і пристроїв, а також проводити purple-team вправи для перевірки здатності системи виявляти складні техніки (наприклад, low-and-slow beaconing або lateral movement із використанням легітимних протоколів). Інтеграція з SIEM і SOAR є критичною: перше забезпечує кореляцію подій із різних джерел (EDR, IAM, хмара), друге дозволяє автоматизувати реагування на виявлені інциденти (ізоляція сегмента, блокування IP чи домену, відкриття інциденту). У комплексі NDR перетворюється з окремого сенсора на стратегічний інструмент для виявлення прихованих атак, що не залишають простих сигнатур, і дає підприємству можливість вчасно виявляти ознаки компрометації навіть у зашифрованому середовищі.

Інтеграція та оркестрація

Максимальна ефективність досягається, коли всі компоненти працюють разом: EDR дає “хлібні крихти” по процесах і командних рядках, NDR — мережевий контекст, IDS/IPS — швидке блокування на периметрі, SIEM — кореляція і історичний аналіз. Типовий робочий процес SOC: NDR/IDS сигналізує про підозрілу мережеву сесію → SIEM корелює подію з нещодавніми EDR-логами → аналітик або SOAR-плейбук ініціює ізоляцію кінцевої точки (EDR) та відкриває тикет на patching/подальше розслідування. Для прискорення цього процесу важливі стандартизовані

формати (CIF/CEF, Elastic Common Schema), правила Sigma/YARA для переносності детекцій, а також TIP-інтеграція для автоматичного збагачення IOC.

Інтеграція та оркестрація систем моніторингу й реагування забезпечують перехід від розрізнених інструментів до єдиного узгодженого середовища захисту. SIEM, EDR, IDS/IPS та NDR повинні бути взаємопов'язані через SOAR-платформи, які автоматизують обробку подій та прискорюють реагування. Такий підхід дозволяє будувати наскрізні playbook-и: наприклад, спрацювання правила в IDS може автоматично створити інцидент у SIEM, перевірити наявність IoC у Threat Intelligence, ізолювати підозрілий хост через EDR і зафіксувати всі дії у системі кейс-менеджменту. Важливим аспектом є уніфікація форматів даних (STIX/TAXII, Sigma, OpenIOC), що полегшує обмін між системами та зменшує кількість ручних операцій. Оркестрація також включає інтеграцію з ITSM-системами (ServiceNow, Jira), CMDB та інвентаризацією активів, що забезпечує контекстність інцидентів і правильну пріоритизацію реагування. У результаті SOC отримує можливість не лише швидко виявляти та локалізувати атаки, але й підтримувати повний життєвий цикл реагування — від виявлення до lessons learned — у єдиному автоматизованому потоці.

Налаштування, тюнінг і управління шумом

Одне з найбільших практичних завдань — зменшення фальшпозитивів при збереженні чутливості. Це досягається через: контекстну нормалізацію (asset criticality, maintenance windows), whitelist/allowlist і білий лист періодичних легітимних патернів, адаптивні правила з порогами на основі baseline, використання аналітики поведінки замість простих порогів і регулярні reviews правил. Метрики якості роботи SOC: MTTD (mean time to detect), MTTR (mean time to remediate), кількість false positives на аналітика, coverage % (частка хостів з EDR/NDR агенцією) і time-to-containment.

Налаштування та тюнінг систем моніторингу потребують постійного циклу ітерацій, оскільки загрозовий ландшафт і сама інфраструктура динамічно змінюються. Управління шумом передбачає не лише зменшення фальшпозитивів, але й баланс між чутливістю та стабільністю роботи SOC. Для цього впроваджуються багаторівневі playbook-и, які дозволяють відсікати низькопріоритетні події, автоматично збагачувати контекст (хто власник активу, яка його критичність, чи знаходиться він у тестовому середовищі), а також застосовувати механізми адаптивної кореляції — наприклад, підвищувати пріоритет подій у неробочий час чи для VIP-акаунтів. Практика включає створення blackhole-правил для відомих легітимних шумових патернів (наприклад, регулярні бекапи, автоматичні

скрипти) та періодичні аудит-сесії SOC, на яких аналізуються тригери з найвищим відсотком фальшспрацьовувань. Крім того, інтеграція з ML/UEBA дозволяє поступово замінювати статичні порогові правила динамічними профілями, що зменшує обсяг ручної роботи аналітиків. Керування шумом розглядається як безперервний процес maturity management, де якісні метрики (MTTD, MTTR, precision/recall детекцій, відсоток автоматизованих containment-дій) стають базою для оцінки ефективності SOC і для стратегічних рішень щодо оптимізації ресурсів.

Практичні аспекти використання систем моніторингу та виявлення атак

Практичні рекомендації щодо використання IDS/IPS, SIEM, EDR і NDR зводяться до побудови багаторівневої та узгодженої стратегії кіберзахисту. Першим кроком має стати повне покриття телеметрією: усі критично важливі сервери та робочі станції повинні перебувати під контролем EDR, периметр і дата-центри — під моніторингом NDR та IDS, а всі журнали подій мають централізовано надходити в SIEM для аналізу. Це дозволяє сформувати цілісну картину активності в інфраструктурі та оперативно виявляти аномалії.

Другим ключовим напрямом є інтеграція з threat intelligence-платформами. Збагачення IoC (Indicators of Compromise) у SIEM, NDR і EDR дає змогу автоматично пріоритизувати події та відокремлювати справді небезпечні індикатори від другорядних. Це особливо важливо за умов великого потоку логів, коли вручну відфільтрувати всі сигнали практично неможливо.

Третім кроком має стати впровадження автоматизації реагування. Для типових сценаріїв компрометації розробляються playbook-и у SOAR: ізоляція інфікованого хоста, блокування підозрілого домену чи IP-адреси, створення інциденту в системі управління безпекою. Такий підхід скорочує час реагування та мінімізує ризик поширення атаки.

Четвертий практичний аспект — постійний fine-tuning і розвиток навичок SOC-команди. Регулярні table-top вправи та тренування purple-team дають змогу перевірити, наскільки коректно працюють правила детекції, чи належно налаштовані джерела телеметрії та чи здатні аналітики ефективно реагувати на реальні атаки.

П'ятий елемент — контроль за приватністю та відповідністю регуляторним вимогам. Зокрема, TLS inspection і довготривале зберігання PCAP-трафіку мають здійснюватися у суворій відповідності до політик безпеки та законодавства. Це допомагає уникнути юридичних ризиків і зберегти довіру користувачів.

Водночас слід враховувати обмеження та ризики. Жодна система не є універсальним захистом: сигнатурні IDS/IPS пропускають zero-day та поліморфні атаки; EDR залежить від агента й може бути обійдений досвідченими зловмисниками; NDR втрачає ефективність унаслідок повсюдного шифрування трафіку без можливості TLS-інспекції; SIEM без належної оптимізації може створювати надмірний «шум», що перевантажує SOC. Тому найбільш надійною стратегією є defense-in-depth — багаторівневий підхід із регулярною перевіркою ефективності та постійною адаптацією до змін у ландшафті загроз.

У підсумку IDS/IPS, SIEM, EDR і NDR становлять чотири взаємодоповнюючі виміри видимості, які у комплексі забезпечують можливість не лише ідентифікації відомих загроз, а й виявлення поведінкових аномалій, оперативного реагування та проведення форензичних розслідувань. Грамотне розгортання, інтеграція та системне налаштування цих рішень перетворює телеметрію на дієвий інструмент для скорочення часу перебування атакувальника в системі та підвищення загальної кіберстійкості підприємства.

Метрики ефективності та оцінювання SOC

Метрики ефективності та оцінювання SOC відіграють ключову роль у вимірюванні зрілості та результативності процесів безпеки. Базовими показниками є MTTD (Mean Time to Detect) — середній час виявлення загрози, та MTTR (Mean Time to Respond/Remediate) — середній час реагування або усунення. Додатково оцінюється TTI (Time to Investigate) — час від моменту виявлення до завершення аналізу інциденту, а також MTTC (Mean Time to Containment) — середній час локалізації атаки. Важливими є й якісні метрики: false positive rate (частка фальшпозитивів), true positive rate, coverage % (покриття EDR/NDR/логами критичних активів), automation rate (відсоток інцидентів, закритих автоматично через SOAR) і analyst workload (кількість подій/інцидентів на одного аналітика). SOC також оцінюють за стратегічними показниками — наприклад, dwell time (середній час перебування атакувальника в системі до виявлення), відповідність регуляторним вимогам (ISO/IEC 27001, NIST CSF, GDPR) і результативність purple-team/penetration-test вправ. Для комплексної оцінки ефективності застосовуються maturity-моделі (SOC-CMM, NIST CSF), які дозволяють визначити рівень зрілості SOC у вимірах процесів, технологій і персоналу. Регулярний моніторинг цих метрик забезпечує не лише прозорість для керівництва, але й формує основу для прийняття інвестиційних рішень, підвищення ефективності автоматизації та зменшення ризиків для бізнесу.

СПИСОК ДЖЕРЕЛ РОЗДІЛУ 3

1. Костюк Ю.В., Складанний П.М., Бебешко Б.Т., Хорольська К.В., Рзаєва С.Л., Ворохоб М.В. (2025) Безпека інформаційно-комунікаційних систем. Київський університет імені Бориса Грінченка, Україна, Київ. - 1017 с.
2. Костюк Ю.В., Складанний П.М., Гулак Г.М., Бебешко Б.Т., Хорольська К.В., Рзаєва С.Л. (2025) Системи захисту інформації. Київський університет імені Бориса Грінченка, Україна, Київ. - 886с.
3. Kostiuk, Yu. V., Skladannyi, P. M., Bebeshko, B. T., Khorolska, K. V., Rzaieva, S. L., & Vorokhob, M. V. (2025). Information and communication systems security. [Textbook] Kyiv: Borys Grinchenko Kyiv Metropolitan University.
4. Kostiuk, Yu. V., Skladannyi, P. M., Hulak, H. M., Bebeshko, B. T., Khorolska, K. V., & Rzaieva, S. L. (2025). Information security systems. [Textbook] Kyiv: Borys Grinchenko Kyiv Metropolitan University.
5. Гулак Г. М., Жильцов О. Б., Киричок Р. В., Коршун Н. В., Складанний П. М. Інформаційна та кібернетична безпека підприємства : підруч. / Г. М. Гулак, О. Б. Жильцов, Р. В. Киричок, Н. В. Коршун, П. М. Складанний – Львів : Видавець Марченко Т. В., 2023. – 370 с.
6. Kuzminykh, I., Ghita, B., Sokolov, V., & Bakhshi, T. (2021). Information Security Risk Assessment. *Encyclopedia*, 1(3), 602-617. <https://doi.org/10.3390/encyclopedia1030050>
7. TajDini, M., Sokolov, V., Kuzminykh, I., Shiaeles, S., & Ghita, B. (2020). Wireless Sensors for Brain Activity—A Survey. *Electronics*, 9(12), 2092. <https://doi.org/10.3390/electronics9122092>
8. Grechaninov, Viktor and Hulak, Hennadii and Hulak, Evgen and Skladannyi, Pavlo and Sokolov, Volodymyr (2021) *Decentralized Access Demarcation System Construction in Situational Center Network Cybersecurity Providing in Information and Telecommunication Systems II* 2021, 3188 (2). pp. 197-206. ISSN 1613-0073
9. Buriachok, Volodymyr and Ageyev, Dmytro and Zhylytsov, Oleksii and Skladannyi, Pavlo and Sokolov, V. Y. (2020) *Invasion Detection Model using Two-Stage Criterion of Detection of Network Anomalies* Cybersecurity Providing in Information and Telecommunication Systems, 2746. pp. 23-32. ISSN 1613-0073
10. Костюк Ю., Хорольська, К., Бебешко, Б., Довженко, Н., Коршун, Н., Пазинін, А. Інструментальні засоби забезпечення

інформаційної безпеки від прихованих загроз в інфраструктурі хмарних обчислень. *Кібербезпека: освіта, наука, техніка*. 2025. 4 (28), 633–655.

11. Складанний, П.М., Гулак, Г.М., Костюк, Ю.В. Генератор хаотичних чисел із нечітким керуванням для криптографічних систем із динамічною довірою. *Телекомунікаційні та інформаційні технології*. 2025. № 4 (89), 137-147. <https://doi.org/10.31673/2412-4338.2025.048916>

12. M. Vladymyrenko, V. Sokolov, V. Buriachok, A. Platonenko and D. Ageyev, "Analysis of Implementation Results of the Distributed Access Control System," *2019 IEEE International Scientific-Practical Conference Problems of Infocommunications, Science and Technology (PIC S&T)*, Kyiv, Ukraine, 2019, pp. 1-6, doi: 10.1109/PICST47496.2019.9061376.

13. Складанний, П.М., Машкіна, І.В., Рзаєва, С.Л., Костюк, Ю.В. Методи GDPR для забезпечення безпеки сховищ даних від витоків та загроз. *Телекомунікаційні та інформаційні технології*. 2025. (2), 59-76.

14. Kostiuk, Yuliia. (2025). Multi-Agent System for Detecting and Counteracting Attacks on the Enterprise Information System. In book: *Insider Threats and Security in Corporations*. pp.205-232. <https://doi.org/10.36690/ITSC-205-232>.

15. Hu, Z., Buriachok, V., Bogachuk, I., Sokolov, V., Ageyev, D. (2021). Development and Operation Analysis of Spectrum Monitoring Subsystem 2.4–2.5 GHz Range. In: Radivilova, T., Ageyev, D., Kryvinska, N. (eds) *Data-Centric Business and Applications. Lecture Notes on Data Engineering and Communications Technologies*, vol 48. Springer, Cham. https://doi.org/10.1007/978-3-030-43070-2_29

16. F. Kipchuk, V. Sokolov, P. Skladannyi and D. Ageyev, "Assessing Approaches of IT Infrastructure Audit," *2021 IEEE 8th International Conference on Problems of Infocommunications, Science and Technology (PIC S&T)*, Kharkiv, Ukraine, 2021, pp. 213-217, doi: 10.1109/PICST54195.2021.9772181.

17. Grechaninov, Viktor and Hulak, Hennadii and Sokolov, V. Y. and Skladannyi, Pavlo and Korshun, Natalia (2022) *Formation of Dependability and Cyber Protection Model in Information Systems of Situational Center* Emerging Technology Trends on the Smart Industry and the Internet of Things 2022, 3149. pp. 107-117. ISSN 1613-0073

18. Костюк, Ю., Складанний, П., Рзаєва, С., Мазур, Н., Черевик, В., & Аносов, А. (2025). Особливості реалізації мережевих атак через TCP/IP-протоколи. *Електронне фахове наукове видання «Кібербезпека: освіта, наука, техніка»*, 1(29), 571–597.

19. Kuzminykh, I., Carlsson, A., Yevdokymenko, M., Sokolov, V. (2019). Investigation of the IoT Device Lifetime with Secure Data Transmission. In: Galinina, O., Andreev, S., Balandin, S., Koucheryavy, Y. (eds) *Internet of*

Things, Smart Spaces, and Next Generation Networks and Systems. NEW2AN ruSMART 2019 2019. Lecture Notes in Computer Science(), vol 11660. Springer, Cham. https://doi.org/10.1007/978-3-030-30859-9_2

20. Buriachok, Volodymyr and Sokolov, V. Y. and Skladannyi, Pavlo (2019) *Security Rating Metrics for Distributed Wireless Systems* MoMLet&DS. pp. 222-233. ISSN 1613-0073

21. Rzaeva S., Skladannyi P., Kostiuk Y., Abramov V., Kravchenko V. Adaptive Information Security Management in Cloud-Oriented Intelligent Transportation Systems. Ukrainian Scientific Journal of Information Security, 2025, vol. 31, issue 1, pp. 23-36. <https://doi.org/10.18372/2225-5036.31.20634>

22. Hu, Z.B., Buriachok, V., TajDini, M., Sokolov, V. (2021). Authentication System by Human Brainwaves Using Machine Learning and Artificial Intelligence. In: Hu, Z., Petoukhov, S., Dychka, I., He, M. (eds) *Advances in Computer Science for Engineering and Education IV. ICCSEEA 2021. Lecture Notes on Data Engineering and Communications Technologies*, vol 83. Springer, Cham. https://doi.org/10.1007/978-3-030-80472-5_31

23. Костюк Ю., Бебешко, Б., Крючкова, Л., Литвинов, В., Оксанич, І., Складанний, П., Хорольська, К. Захист інформації та безпека обміну даними в безпроводових мобільних мережах з автентифікацією і протоколами обміну ключами. Кібербезпека: освіта, наука, техніка. 2024. 1(25), 229–252.

24. Buhas, Vasyl and Ponomarenko, Ihor and Bugas, Valeriy and Ramskyi, Andrii and Sokolov, Volodymyr (2021) *Using Machine Learning Techniques to Increase the Effectiveness of Cybersecurity* Cybersecurity Providing in Information and Telecommunication Systems II 2021, 3188 (2). pp. 273-281. ISSN 1613-0073

25. Zhebka, V. and Skladannyi, Pavlo and Zhebka, S. and Shlianachak, S. and Bondarchuk, A. (2024) *Methodology for Predicting Failures in a Smart Home based on Machine Learning Methods* Cybersecurity Providing in Information and Telecommunication Systems 2024, 3654. pp. 322-332. ISSN 1613-0073

26. Skladannyi, Pavlo and Trofimov, Olexandr and Korniiets, Viktor and Vorokhob, Maksym and Opryshko, Tetiana (2023) *Improving the Security Policy of the Distance Learning System based on the Zero Trust Concept* CPITS 2023: Workshop on Cybersecurity Providing in Information and Telecommunication Systems. pp. 97-106. ISSN 1613-0073

27. Довженко, Н., Іваніченко, Є., & Костюк, Ю. (2025). Методика виявлення та локалізації кіберзагроз у хмарних середовищах з інтегрованими ІОТ-компонентами на основі графових моделей. Електронне

фахове наукове видання «Кібербезпека: освіта, наука, техніка», 1(29), 762–776.

28. Шевченко, Галина Володимирівна and Шевченко, Світлана Миколаївна and Жданова, Юлія Дмитрівна and Спасітелєва, Світлана Олексіївна and Негоденко, Олена Василівна (2021) *Information Security Risk Analysis SWOT Cybersecurity Providing in Information and Telecommunication Systems*, 2923. pp. 309-317. ISSN 1613-0073

29. I. Iosifov, O. Iosifova and V. Sokolov, "Sentence Segmentation from Unformatted Text using Language Modeling and Sequence Labeling Approaches," *2020 IEEE International Conference on Problems of Infocommunications. Science and Technology (PIC S&T)*, Kharkiv, Ukraine, 2020, pp. 335-337, doi: 10.1109/PICST51311.2020.9468084.

30. Petriv, P. and Oprisky, I. and Mazur, Nataliia (2024) *Modern technologies of decentralized databases, authentication, and authorization methods Cybersecurity Providing in Information and Telecommunication Systems II* 2024, 3826. pp. 60-71. ISSN 1613-0073

31. Astafieva, Mariia and Bodnenko, Dmitry and Lytvyn, Oksana and Proshkin, Volodymyr and Skladannyi, Pavlo (2023) *Formation of High School Students' Resistance to Destructive Information Influences Cybersecurity Providing in Information and Telecommunication Systems* 2023, 3421. pp. 87-96. ISSN 1613-0073

32. Sovyn, Y. and Khoma, V. and Oprisky, I. and Kozachok, Valerii (2023) *Minimization of Bitsliced Representation of 4×4 S-Boxes based on Ternary Logic Instruction Cybersecurity Providing in Information and Telecommunication Systems* 2023, 3421. pp. 12-24. ISSN 1613-0073

33. Burov, O. et al. (2020). Cybersecurity in Educational Networks. In: Ahram, T., Karwowski, W., Vergnano, A., Leali, F., Taiar, R. (eds) *Intelligent Human Systems Integration 2020. IHSI 2020. Advances in Intelligent Systems and Computing*, vol 1131. Springer, Cham. https://doi.org/10.1007/978-3-030-39512-4_56

34. Buriachok, V., Sokolov, V. (2020). Implementation of Active Learning in the Master's Program on Cybersecurity. In: Hu, Z., Petoukhov, S., Dychka, I., He, M. (eds) *Advances in Computer Science for Engineering and Education II. ICCSEEA 2019. Advances in Intelligent Systems and Computing*, vol 938. Springer, Cham. https://doi.org/10.1007/978-3-030-16621-2_57

35. Anakhov, Pavlo and Zhebka, Viktoriia and Koretska, Viktoriia and Sokolov, V. Y. and Skladannyi, Pavlo (2022) *Increasing the Functional Network Stability in the Depression Zone of the Hydroelectric Power Station Reservoir Emerging Technology Trends on the Smart Industry and the Internet of Things*, 3149. pp. 169-176. ISSN 1613-0073

36. Z. Hu, B. Vovkotrub, V. Buriachok, Y. Zotkin and V. Sokolov, "Bandwidth Research of Wireless IoT Switches," *2020 IEEE 15th International Conference on Advanced Trends in Radioelectronics, Telecommunications and Computer Engineering (TCSET)*, Lviv-Slavske, Ukraine, 2020, pp. 1-5, doi: 10.1109/TCSET49122.2020.235492.
37. Bessalov, Anatoly and Sokolov, Volodymyr and Skladannyi, Pavlo and Abramov, Serhii and Zhylytsov, Oleksii (2022) *Modeling CSIKE Algorithm on Non-Cyclic Edwards Curves* Cybersecurity Providing in Information and Telecommunication Systems 2022, 3288 (1). pp. 1-10. ISSN 1613-0073
38. Рекомендації CERT-UA щодо протидії АРТ (Advanced Persistent Threat). URL: <https://cert.gov.ua/recommendations/18>
39. Технологія VPN може загрожувати кібербезпеці. URL: <https://detector.media/infospace/article/126505/2017-05-31-tekhnologiya-vpnmozhe-zagrozhuvati-kiberbezpetsi-inau>
40. JP 3–12 Cyberspace Operations, 8 June 2018. URL: https://www.jcs.mil/Portals/36/Documents/Doctrine/pubs/jp3_12.pdf
41. DOD Dictionary of Military and Associated Terms. As
42. of January 2019. URL: <https://www.jcs.mil/Portals/36/Documents/Doctrine/pubs/dictionary.pdf>
43. Cybersecurity a generic reference curriculum (2016). URL: https://www.nato.int/nato_static_fl2014/assets/pdf/pdf_2016_10/1610-cybersecurity-162_curriculum.pdf
44. National Cyber Security Strategy Good Practice Guide (2016). URL: <https://www.enisa.europa.eu/publications/ncss-good-practice-guide>
45. Castro D. (2018). Boosting the Cyberworkforce. URL: <http://www.govtech.com/data/Boosting-the-Cyberworkforce.html>
46. Cisco -Україна. URL: <https://www.cisco.com>
47. Bessalov, Anatoly and Sokolov, Volodymyr and Skladannyi, Pavlo and Mazur, Nataliia and Ageyev, Dmytro (2021) *Implementation of the CSIDH Algorithm Model on Supersingular Twisted and Quadratic Edwards Curves* Cybersecurity Providing in Information and Telecommunication Systems II 2021, 3187 (1). pp. 302-309. ISSN 1613-0073
48. Bessalov, Anatoly and Kovalchuk, L. and Sokolov, V. Y. and Skladannyi, Pavlo and Radivilova, T. (2020) *Analysis of 2-Isogeny Properties of Generalized Form Edwards Curves* Cybersecurity Providing in Information and Telecommunication Systems, 2746. pp. 1-13. ISSN 1613-0073
49. Bessalov, Anatolii & Grubiyan, Evgeniy & Sokolov, Volodymyr & Skladannyi, Pavlo. (2020). 3- and 5-Isogenies of Supersingular Edwards Curves. 10.48550/arXiv.2006.16111.

-
50. Bessalov, Anatoly and Sokolov, V. Y. and Skladannyi, Pavlo and Zhylytsov, Oleksii (2021) *Computing of Odd Degree Isogenies on Supersingular Twisted Edwards Curves* Cybersecurity Providing in Information and Telecommunication Systems, 2923. pp. 1-11. ISSN 1613-0073
51. V. Astapenya, V. Sokolov, P. Skladannyi and O. Zhylytsov, "Analysis of Ways and Methods of Increasing the Availability of Information in Distributed Information Systems," *2021 IEEE 8th International Conference on Problems of Infocommunications, Science and Technology (PIC S&T)*, Kharkiv, Ukraine, 2021, pp. 174-178, doi: 10.1109/PICST54195.2021.9772161.
52. Рзаєва С.Л., Складанний П.М., Машкіна І.В., Костюк Ю.В. (2025) Модель реалізації керування доступом на основі ролей (RBAC) у багаторівневій архітектурі сховища даних. Сучасний захист інформації, 3(63), 137–149.
53. Костюк, Ю., Складанний, П., Рзаєва, С., Самойленко, Ю., & Коршун, Н. (2025). Інтелектуальні системи керування та захисту в кіберфізичних і хмарних середовищах Smart Grid. Електронне фахове наукове видання «Кібербезпека: освіта, наука, техніка», 2(30), 125–156.
54. V. Astapenya, V. Buriachok, V. Sokolov, P. Skladannyi and D. Ageyev, "Last Mile Technique for Wireless Delivery System using an Accelerating Lens," *2020 IEEE International Conference on Problems of Infocommunications. Science and Technology (PIC S&T)*, Kharkiv, Ukraine, 2020, pp. 811-814, doi: 10.1109/PICST51311.2020.9467886.
55. Bessalov, Anatoly and Abramov, Serhii and Sokolov, Volodymyr and Skladannyi, Pavlo and Zhylytsov, Oleksii (2023) *Multifunctional CRS Encryption Scheme on Isogenies of Non-Supersingular Edwards Curves* Workshop on Classic, Quantum, and Post-Quantum Cryptography (CQPC 2023), 3504. pp. 12-25. ISSN 1613-0073
56. TajDini, Mahyar and Sokolov, V. Y. and Kuzminykh, Ievgeniia and Ghita, B. (2023) *Brainwave-based authentication using features fusion* Computers & Security (129). p. 103198. ISSN 0167-4048
57. Iosifova, Olena and Iosifov, Ievgen and Rolik, Oleksandr and Sokolov, V. Y. (2020) *Techniques Comparison for Natural Language Processing* MoMLet&DS, 2631 (I). pp. 57-67. ISSN 1613-0073
58. Anakhov, P. and Zhebka, V. and Tushych, A. and Kravchenko, V. and Blazhennyi, N. and Skladannyi, Pavlo and Sokolov, Volodymyr (2022) *Evaluation Method of the Physical Compatibility of Equipment in a Hybrid Information Transmission Network* Journal of Theoretical and Applied Information Technology, 100 (22). pp. 6635-6644. ISSN 1817-3195
59. Y. Sadykov, V. Sokolov and P. Skladannyi, "Technology of Location Hiding by Spoofing the Mobile Operator IP Address," *2021 IEEE International*

Conference on Information and Telecommunication Technologies and Radio Electronics (UkrMiCo), Odesa, Ukraine, 2021, pp. 22-25, doi: 10.1109/UkrMiCo52950.2021.9716700.

60. Obushnyi, Sergiy and Virovets, Denis and Hulak, Hennadii and Zhurakovskiy, Bohdan (2022) *Autonomy of Economic Agents in Peer-to-Peer Systems Cybersecurity Providing in Information and Telecommunication Systems* 2022, 3288 (1). pp. 125-133. ISSN 1613-0073

61. TajDini, Mahyar and Sokolov, V. Y. and Buriachok, Volodymyr (2019) *Men-in-the-Middle Attack Simulation on Low Energy Wireless Devices using Software Define Radio MoMLeT&DS* (2386). pp. 287-296. ISSN 1613-0073

62. V. Sokolov, P. Skladannyi and A. Platonenko, "Video Channel Suppression Method of Unmanned Aerial Vehicles," *2022 IEEE 41st International Conference on Electronics and Nanotechnology (ELNANO)*, Kyiv, Ukraine, 2022, pp. 473-477, doi: 10.1109/ELNANO54667.2022.9927105.

63. Bessalov, Anatoliy and Abramov, Serhii and Sokolov, Volodymyr and Mazur, Nataliia (2023) *CSIKE-ENC Combined Encryption Scheme with Optimized Degrees of Isogeny Distribution Cybersecurity Providing in Information and Telecommunication Systems*, 3421. pp. 36-45. ISSN 1613-0073

64. Складаний, П., Костюк, Ю., Рзаєва, С., & Мазур, Н. (2025). *Паралельна обробка даних у розширюваних хеш-структурах та оцінка їх продуктивності*. Електронне фахове наукове видання «Кібербезпека: освіта, наука, техніка», 3(31), 242–269. <https://doi.org/10.28925/2663-4023.2025.31.1015>

КОНТРОЛЬНІ ПИТАННЯ

1. Які основні категорії атакуювальників виокремлюєте в моделях загроз і за якими ознаками вони відрізняються (мотиви, ресурси, TTPs)?

2. Чому supply-chain атаки є критично небезпечними для організацій і які контрольні заходи проти них варто впроваджувати?

3. Які відмінності між DDoS на L3/L4 і L7, та як змінюються підходи до їхнього виявлення і мітимації?

4. У чому сутність ARP-spoofing та які механізми L2-рівня (DAI, порт-безпека) і мережеві політики допомагають його блокувати?

5. Поясніть роль BCP38/uRPF у протидії IP-spoofing. Чому ці практики важливі саме на краю мережі та у провайдера?

6. Які індикатори/телеметрія (NetFlow/IPFIX, PCAP, TLS-fingerprinting) найбільш корисні для раннього виявлення мережевих атак і чому?

7. Розкрийте механіку TCP hijacking та сучасні причини успіху «захоплень» через сесійні токени. Які контрзаходи дієвіші за класичні?
8. Чим UDP-flooding із ампліфікацією відрізняється від «чистого» UDP-флуду? Наведіть приклади сервісів-рефлекторів і способи їхнього харденінгу.
9. Поясніть, як працює SYN-cookies і чому це ключовий метод проти SYN-flood. Які є додаткові мережеві прийоми?
10. Порівняйте SQLi, XSS, CSRF та Command Injection за джерелом помилки, вектором атаки і наслідками. Які «базові» DevSecOps-практики закривають більшість цих ризиків?
11. Які політики та техніки (CSP, HttpOnly/Secure/SameSite, санітайзери) знижують ризик XSS у сучасних SPA/GraphQL-архітектурах?
12. Як правильно проєктувати захист від CSRF у поєднанні з CORS і токен-базованою автентифікацією, щоб уникнути колізій у захисних механізмах?
13. Які принципи й інструменти слід застосовувати для попередження Command Injection у скриптах CI/CD та системних викликах застосунків?
14. Поясніть різницю між SSL-VPN, IPsec-VPN і WireGuard: де який підхід виправданий з погляду продуктивності, зручності та криптографії?
15. Які сучасні налаштування IPsec (IKEv2/MOBIKE, AEAD-шифри, NAT-T, апаратні прискорювачі) критично впливають на стійкість і масштабованість?
16. Для чого організації використовують MPLS-VPN, і чому часто комбінують MPLS із SD-WAN та/або IPsec поверх MPLS?
17. Що таке ZTNA і як воно інтегрується в SASE-архітектуру? Наведіть приклади контекстних сигналів, які впливають на рішення про доступ.
18. Які класи засобів спостережності/детекції (EDR, NDR, SIEM, UEBA, SOAR) формують «нервову систему» захисту? Як між собою корелюють їхні сигнали?
19. Які специфічні виклики безпеки притаманні SDN (захист контролера, southbound/northbound API), і як принцип Zero Trust переноситься в SDN/NFV?
20. Які ключові метрики (TTD, TTC, покриття MITRE ATT&CK) ви б застосували для вимірювання ефективності мережевої безпеки та як їх покращувати на практиці?

РОЗДІЛ 4

ІНСТРУМЕНТИ ТА ТЕХНОЛОГІЇ ЗАБЕЗПЕЧЕННЯ ІНФОРМАЦІЙНОЇ БЕЗПЕКИ: СИСТЕМИ КОНТРОЛЮ ДОСТУПУ, БРАНДМАУЕРИ, IDS/IPS, SIEM/SOAR ТА МОНІТОРИНГ ПОДІЙ



- 4.1. Моделі контролю доступу (DAC, MAC, RBAC, ABAC, Zero Trust)**
- 4.2. Міжмережеві екрани (статичні, динамічні, NGFW) та їх роль у багаторівневому захисті**
- 4.3. Системи виявлення та запобігання вторгнень (IDS/IPS, HIDS, NIDS)**
- 4.4. Інструменти SIEM та SOAR для централізованого моніторингу і реагування**
- 4.5. Засоби захисту кінцевих точок: антивірусні системи, EDR/XDR**

Інструменти та технології забезпечення інформаційної безпеки відіграють ключову роль у захисті сучасних інформаційно-комунікаційних систем. Основою є системи контролю доступу, які визначають рівні прав користувачів, реалізують принцип мінімальних привілеїв і концепцію Zero Trust, що мінімізує ризики несанкціонованого проникнення. Брандмауери створюють бар'єр між внутрішньою мережею та зовнішнім середовищем, фільтруючи трафік відповідно до заданих правил і запобігаючи проникненню небезпечних пакетів даних. Для більш глибокого захисту застосовуються IDS/IPS-системи, які аналізують мережеву активність: IDS

дозволяє виявляти атаки та повідомляти про них, а IPS забезпечує активне блокування шкідливих дій у реальному часі. Важливим компонентом є SIEM-платформи, що централізовано збирають, корелюють і аналізують журнали подій безпеки, забезпечуючи повну картину стану захисту, тоді як рішення SOAR автоматизують реагування на інциденти та оркеструють взаємодію між різними засобами безпеки. Завершальним елементом є постійний моніторинг подій, який дозволяє в режимі реального часу відстежувати відхилення у роботі систем, оперативно реагувати на інциденти та підтримувати безперервність кіберзахисту. Таким чином, поєднання цих технологій формує багаторівневу та ефективну систему протидії сучасним кіберзагрозам.

4.1. МОДЕЛІ КОНТРОЛЮ ДОСТУПУ (DAC, MAC, RBAC, ABAC, ZERO TRUST)

Моделі контролю доступу є фундаментом забезпечення інформаційної безпеки, оскільки визначають правила взаємодії користувачів із ресурсами системи та рівень довіри між ними. Кожна модель має свої особливості, сферу застосування та обмеження, а сучасна практика часто поєднує кілька підходів для досягнення оптимального рівня захисту. Вибір конкретної моделі залежить від масштабів підприємства, типу інформації та рівня ризику, який потрібно врахувати. У сучасних інформаційно-комунікаційних системах усе частіше застосовується багаторівневий підхід, де моделі комбінуються. Це дозволяє компенсувати недоліки одних рішень перевагами інших та досягати більшої стійкості до атак.

Моделі контролю доступу є фундаментом забезпечення інформаційної безпеки, оскільки визначають правила взаємодії користувачів із ресурсами системи та рівень довіри між ними. Кожна модель має свої особливості, сферу застосування та обмеження, а сучасна практика часто поєднує кілька підходів для досягнення оптимального рівня захисту. Вибір конкретної моделі залежить від масштабів підприємства, типу інформації та рівня ризику, який потрібно врахувати. У сучасних інформаційно-комунікаційних системах усе частіше застосовується багаторівневий підхід, де моделі комбінуються. Це дозволяє компенсувати недоліки одних рішень перевагами інших та досягати більшої стійкості до атак. Додатково важливим чинником стає автоматизація управління доступом за допомогою штучного інтелекту та аналітики поведінки користувачів. Такі інструменти дають змогу виявляти аномалії у реальному часі й адаптивно коригувати політики безпеки. Поєднання класичних і новітніх моделей формує основу для створення сучасних архітектур захисту, орієнтованих на принципи Zero Trust. Це забезпечує не лише технічну безпеку, а й відповідність міжнародним стандартам та регуляторним вимогам.

DAC (Discretionary Access Control)

Сучасні дослідження відзначають, що DAC усе ще залишається актуальною в невеликих системах або середовищах, де пріоритетом є простота управління. Однак, через відсутність централізованого контролю, ця модель часто стає ціллю атак із застосуванням соціальної інженерії. Щоб зменшити ризики, у практиці її поєднують із моніторингом поведінки користувачів і додатковими механізмами виявлення аномалій. DAC зручна для стартапів, навчальних середовищ чи малих підприємств, але не відповідає сучасним вимогам критичної інфраструктури. Вона дозволяє швидко змінювати доступ, однак не гарантує від внутрішніх загроз. Використання DAC у поєднанні з MFA та аудитом може частково підвищити її ефективність.

Історично DAC стала однією з перших моделей контролю доступу та реалізована в більшості операційних систем, включаючи UNIX і Windows, де власник файлу сам визначає права доступу. Проте слабким місцем є можливість поширення шкідливого ПЗ через облікові записи користувачів, яким власник випадково або навмисне надав надмірні дозволи. Для зменшення цього ризику у сучасних рішеннях застосовують принцип least privilege (мінімально необхідних прав), який обмежує доступ до критичних ресурсів навіть у рамках DAC. Крім того, системні адміністратори дедалі частіше інтегрують DAC із SIEM-платформами для виявлення аномалій і побудови повної картини активності користувачів.

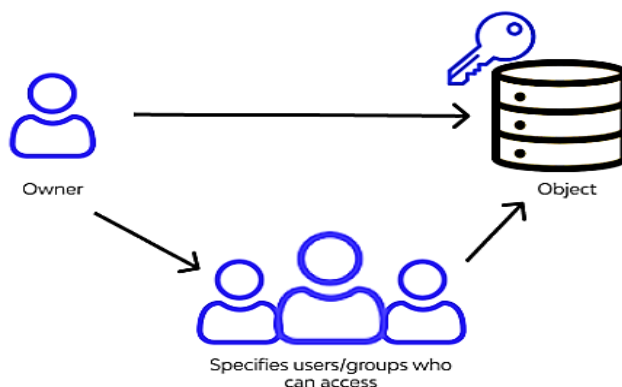


Рис. 4.1. Дискреційна модель контролю доступу (DAC)

Джерело: URL: <https://www.netacad.com/courses/cybersecurity/>

Рис. 4.1. ілюструє принцип роботи дискреційної моделі контролю доступу (DAC), у якій ключова роль належить власнику ресурсу. Власник об'єкта, наприклад файлу чи бази даних, самостійно визначає, які користувачі або групи отримають доступ і в якому обсязі. Таким чином, управління правами здійснюється на розсуд власника, що забезпечує гнучкість і простоту використання. Проте такий підхід має недолік, оскільки користувачі можуть передавати свої права іншим, що підвищує ризик зловживань та витоку

інформації. Модель DAC зручна для малих систем і робочих груп, однак не гарантує достатнього рівня захисту для критичних середовищ.

MAC (Mandatory Access Control)

MAC активно інтегрується в сучасні системи з елементами машинного навчання для класифікації рівнів доступу. Новітні рішення дозволяють автоматично визначати чутливість інформації за допомогою алгоритмів штучного інтелекту, що знижує ризик людської помилки. Крім того, MAC широко застосовується у хмарних обчисленнях та середовищах критичної інфраструктури, де потрібен суворий поділ прав і жорстке дотримання політик. Така модель унеможливорює довільні зміни користувачами, оскільки політики визначаються централізовано. Це робить MAC ефективною для військових, урядових і фінансових систем. Водночас вона є складною у впровадженні й вимагає значних витрат на адміністрування. У гібридних середовищах її зазвичай комбінують із RBAC або ABAC.

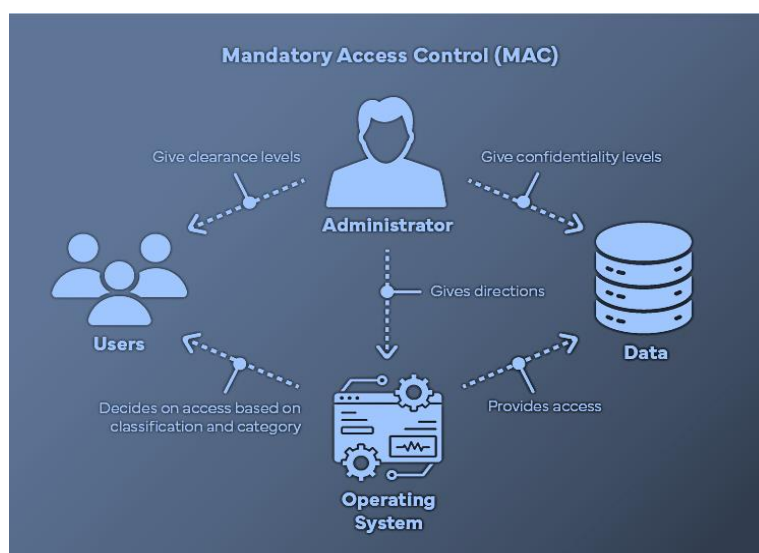


Рис. 4.2. Схема роботи мандатного контролю доступу (MAC)
Джерело: <https://www.netacad.com/courses/cybersecurity/>

На рис. 4.2 показано, як у моделі MAC контроль доступу управляється централізовано через адміністратора, який призначає рівні дозволів (clearance levels) для користувачів і мітки конфіденційності (confidentiality levels) для даних. Далі система (операційна система або механізм контролю доступу) застосовує ці мітки, зіставляючи рівень користувача з рівнем ресурсу, щоб визначити, чи буде доступ дозволено. Принцип полягає в тому, що власники ресурсів не можуть самостійно змінювати політику доступу — права встановлює адміністрація. Таким чином реалізується суворий контроль за

доступом до чутливих об'єктів, зменшуючи ризик витоку інформації чи несанкціонованого доступу, особливо в критично важливих системах.

Історично MAC була розроблена для урядових структур і класифікованих систем, де інформація має мітки безпеки: «таємно», «секретно», «особливо секретно». У таких середовищах рішення про доступ завжди приймається відповідно до співставлення рівня користувача й рівня даних, що мінімізує ризик витоку. Сьогодні MAC застосовується також у віртуалізації та контейнеризації, наприклад у Linux Security Modules (SELinux, AppArmor), які забезпечують обмеження доступу до системних ресурсів. Важливим викликом є баланс між високим рівнем захищеності та гнучкістю бізнес-процесів: надмірно жорсткі політики можуть ускладнювати роботу персоналу. Для мінімізації цього ризику сучасні компанії поєднують MAC із технологіями адаптивного моніторингу та автоматизованого реагування на інциденти.

RBAC (Role-Based Access Control)

Сучасні версії RBAC еволюціонували в напрямку RBAC 2.0, що підтримує динамічні ролі й умовні політики доступу. У великих підприємствах рольова модель інтегрується з SIEM та IAM-системами, забезпечуючи прозорий контроль і аудит. Такий підхід дозволяє швидко виявляти відхилення у поведінці користувачів і оперативно адаптувати політики безпеки до нових вимог. RBAC добре масштабується, що робить її популярною в банківському секторі та IT-компаніях. Вона полегшує onboarding і offboarding співробітників, знижуючи ризики помилок при ручному призначенні доступу. Однак у складних сценаріях із великою кількістю перетинів ролей вона може стати громіздкою. Для підвищення ефективності часто впроваджується у поєднанні з ABAC.

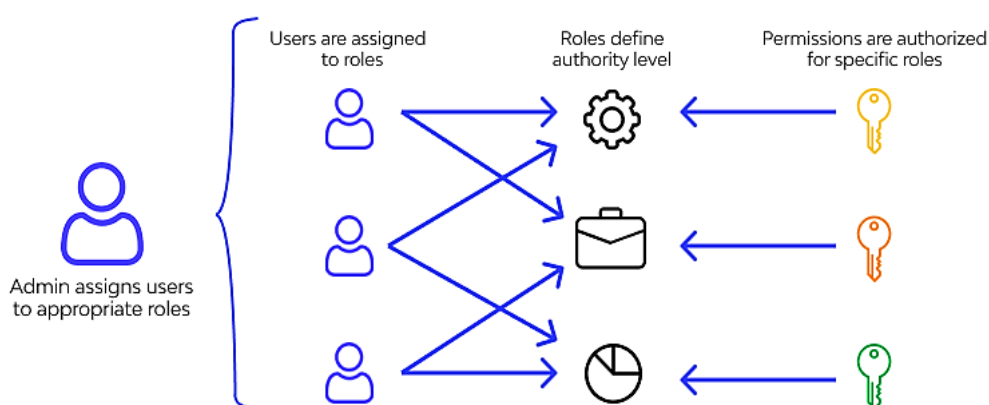


Рис. 4.3. Схема рольового контролю доступу (RBAC)

Джерело: <https://www.netacad.com/courses/cybersecurity/>

На схемі показано, як у моделі RBAC доступ до ресурсів визначається через ролі: користувачі спочатку отримують роль (наприклад, «адміністратор»),

«менеджер», «користувач»), після чого ці ролі мають передані права чи дозволи над певними об'єктами (наприклад, файли, служби, модулі). Кожна роль визначає набір операцій, які може виконувати користувач — читання, запис, видалення тощо — а сам користувач не має прямих індивідуальних дозволів, лише через роль. Схема підкреслює центральне правило RBAC: права не прив'язуються до окремих користувачів, а до ролей, а користувачам присвоюють одну чи кілька ролей. Такий підхід значно спрощує управління доступом у великих системах і знижує ризик помилок при зміні призначень доступу. Історично RBAC стала компромісом між гнучкістю DAC та суворістю MAC, забезпечивши централізований, але водночас зручний контроль доступу.

Сучасні системи підтримують ієрархічні ролі, що дозволяє будувати багаторівневі структури доступу для різних відділів та рівнів управління. Крім того, RBAC активно використовується у хмарних середовищах, де управління доступом до ресурсів здійснюється через ролі користувачів у сервісах AWS, Azure та Google Cloud. Обмеження цієї моделі полягає в тому, що вона не враховує динамічні контекстні фактори (час, місце, пристрій), що робить її менш ефективною проти сучасних загроз. Тому нині вона дедалі частіше комбінується з атрибутивними моделями (ABAC) або Zero Trust для досягнення балансу між безпекою та гнучкістю.

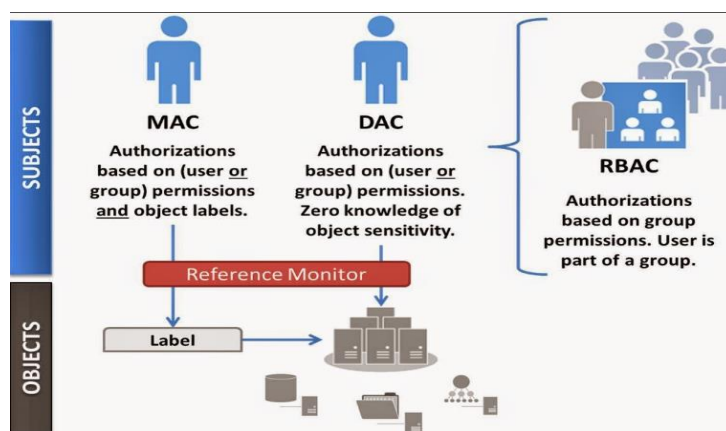


Рис. 4.4. Порівняння моделей контролю доступу (MAC, DAC та RBAC)
Джерело: <https://www.netacad.com/courses/cybersecurity/>

Рис. 4.4 за принципами авторизації» демонструє відмінності між трьома підходами до управління доступом у системах. У моделі MAC (мандатний контроль) доступ визначається централізовано за допомогою міток безпеки та перевіряється спеціальним механізмом Reference Monitor, що забезпечує суворе дотримання політик. У DAC (дискреційний контроль) права встановлює власник ресурсу, що робить модель гнучкою, але вразливою до зловживань. Натомість RBAC (рольова модель) ґрунтується на групових дозволах, де

користувач отримує доступ через належність до певної ролі, що спрощує управління у великих організаціях. Таким чином, рисунок відображає еволюцію від індивідуального керування правами до централізованих і масштабованих підходів, які підвищують ефективність та безпеку систем.

ABAC (Attribute-Based Access Control)

ABAC сьогодні активно використовується у поєднанні з Big Data-аналітикою та системами оцінки ризиків у реальному часі. Рішення про доступ приймаються на основі атрибутів користувача, контексту та рівня загрози. Наприклад, система може автоматично заблокувати сесію, якщо виявлено нетипову геолокацію чи підозрілу поведінку. Це робить ABAC універсальним підходом для сучасних корпоративних середовищ, особливо з віддаленим і мобільним доступом. Його сила полягає у високій гнучкості, що дозволяє враховувати навіть найдрібніші фактори ризику. Водночас складність управління атрибутами вимагає потужних інструментів IAM. ABAC особливо корисний для підприємств, які працюють у багатонаціональних і регульованих середовищах.

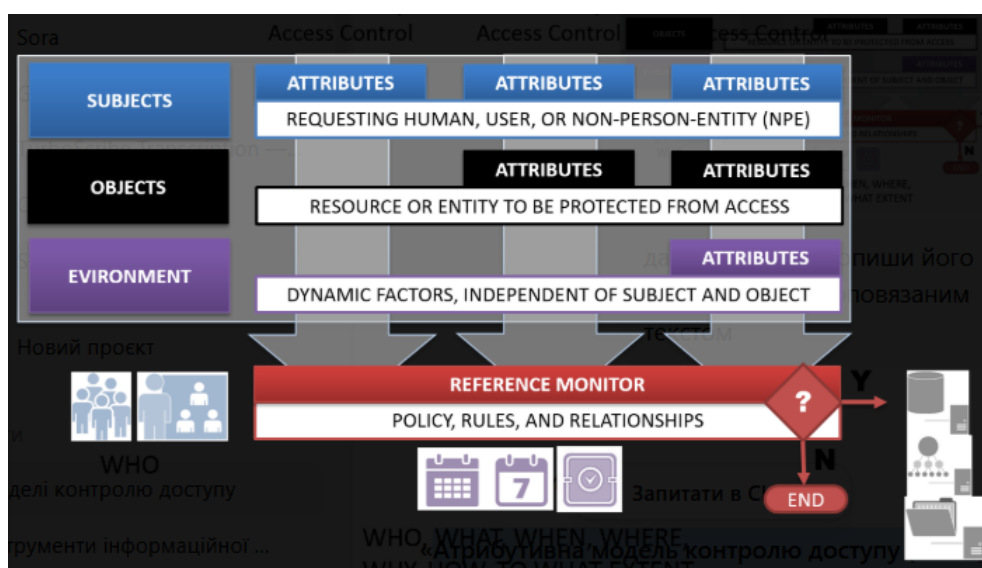


Рис. 4.5. Атрибутивна модель контролю доступу (ABAC): взаємодія суб'єктів, об'єктів та середовища

Джерело: URL: <https://www.netacad.com/courses/cybersecurity/>

Рис. 4.5 демонструє принцип роботи моделі ABAC (Attribute-Based Access Control), у якій рішення про надання доступу приймається на основі атрибутів. У схемі виділено три ключові складові: суб'єкти (користувачі, люди чи цифрові сутності), об'єкти (ресурси, що потребують захисту) та середовище (контекстні умови, незалежні від суб'єкта й об'єкта). Кожен із цих елементів має набір атрибутів — наприклад, роль, відділ чи місце розташування для

користувача; рівень чутливості чи категорія для об'єкта; а також час доби чи геолокація для середовища. Усі ці дані надходять до *Reference Monitor*, який застосовує політики, правила та зв'язки для ухвалення рішення щодо доступу. Таким чином, модель АВАС дозволяє враховувати динамічні фактори та створювати гнучкі контекстно-залежні політики, що значно підвищує ефективність захисту інформаційних систем у сучасних умовах.



Рис. 4.6. Порівняння моделей контролю доступу: RBAC, EBAC та ABAC)
Джерело: URL: <https://www.netacad.com/courses/cybersecurity/>

Рис. 4.6 відображає три різні підходи до контролю доступу в інформаційних системах: рольовий (RBAC), заснований на дозволах (EBAC) та атрибутивний (ABAC). У RBAC користувач отримує права відповідно до своєї ролі, наприклад «адміністратор» чи «менеджер проекту», що спрощує управління у великих організаціях. EBAC ґрунтується на конкретних дозволах, пов'язаних з операціями чи функціями, такими як *user.delete* або *service.edit*, що надає більш точний контроль над діями. ABAC використовує атрибути — наприклад адресу, ім'я чи інші характеристики користувача й середовища — для динамічного визначення доступу. Таким чином, рисунок ілюструє еволюцію від простого рольового керування до більш гнучких і контекстно-залежних моделей доступу.

Історично АВАС виникла як розвиток RBAC, коли стало очевидним, що ролей недостатньо для врахування всіх можливих сценаріїв доступу. На відміну від RBAC, вона дозволяє створювати динамічні політики на основі десятків або сотень змінних, включаючи пристрій, мережу, тип даних, місце розташування та поведінку користувача. У сучасних хмарних рішеннях АВАС інтегрується з Zero Trust-моделлю, забезпечуючи постійний контроль контексту доступу. Проте основним викликом залишається висока складність у проєктуванні правил: при некоректному налаштуванні система може або надмірно

обмежувати роботу співробітників, або створювати «дірки» у безпеці. Для мінімізації цього ризику компанії застосовують машинне навчання для автоматичного визначення політик доступу та прогнозування потенційних загроз.

Zero Trust

Zero Trust дедалі більше визнається міжнародним стандартом завдяки NIST SP 800-207 та рекомендаціям провідних виробників кібербезпеки. Її основний принцип — «ніколи не довіряй, завжди перевіряй» — передбачає безперервну перевірку ідентичності користувача, пристрою й середовища виконання. У практиці ця модель поєднується з багатофакторною аутентифікацією, поведінковою біометрією, алгоритмами виявлення аномалій та автоматизованим реагуванням на інциденти. Zero Trust особливо актуальна для гібридних та мультихмарних інфраструктур, де традиційні периметрові підходи більше неефективні. Вона забезпечує мікросегментацію доступу та контроль за кожною транзакцією. У Zero Trust значна увага приділяється моніторингу поведінки та аналізу ризиків у реальному часі. Компанії, що впроваджують цей підхід, підвищують стійкість до сучасних АРТ-атак і внутрішніх загроз.

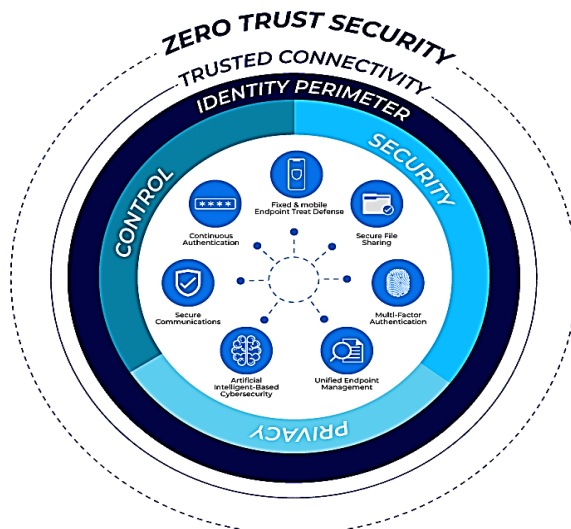


Рис. 4.7. Модель Zero Trust: принцип “ніколи не довіряй, завжди перевіряй”
Джерело: URL: <https://www.netacad.com/courses/cybersecurity/>

На схемі 4.7 зображено центральну ідею Zero Trust, де кожен запит на доступ проходить постійну перевірку незалежно від того, чи походить він зсередини чи ззовні мережі. У центрі — механізм перевірки, який виконує багаторівневе підтвердження ідентичності користувача, пристрою, контексту (час, місце, стан середовища). Далі показано, що доступ до ресурсів (даних, сервісів, інфраструктури) підлягає політикам, які враховують мінімальний привілей (least privilege), мікросегментацію мережі та постійний аналіз ризиків.

Схема демонструє, що навіть «внутрішні» запити не вважаються безпечними за замовчуванням — кожен запит негайно проходить контроль і верифікацію. Такий підхід значно зменшує можливість латерального руху атакуючих всередині мережі та мінімізує площу ураження при компрометації одного з елементів системи.

Історично Zero Trust виникла як відповідь на зростання кількості атак у хмарних середовищах та мобільних мережах, де класичні периметрові моделі не забезпечували захист. На відміну від DAC, MAC чи RBAC, вона не ґрунтується на статичних правилах, а передбачає постійний динамічний аналіз кожного запиту доступу. У сучасних системах Zero Trust інтегрується з XDR, SIEM та SOAR-рішеннями, що дозволяє поєднувати контроль доступу з автоматизованим реагуванням на інциденти. Одним із ключових викликів є складність впровадження: компаніям потрібно перебудовувати архітектуру мережі, створювати політики мікросегментації та налагоджувати взаємодію різних систем захисту. Проте переваги очевидні — скорочення часу виявлення загроз, зниження ризику компрометації облікових даних і підвищення відповідності міжнародним стандартам безпеки.

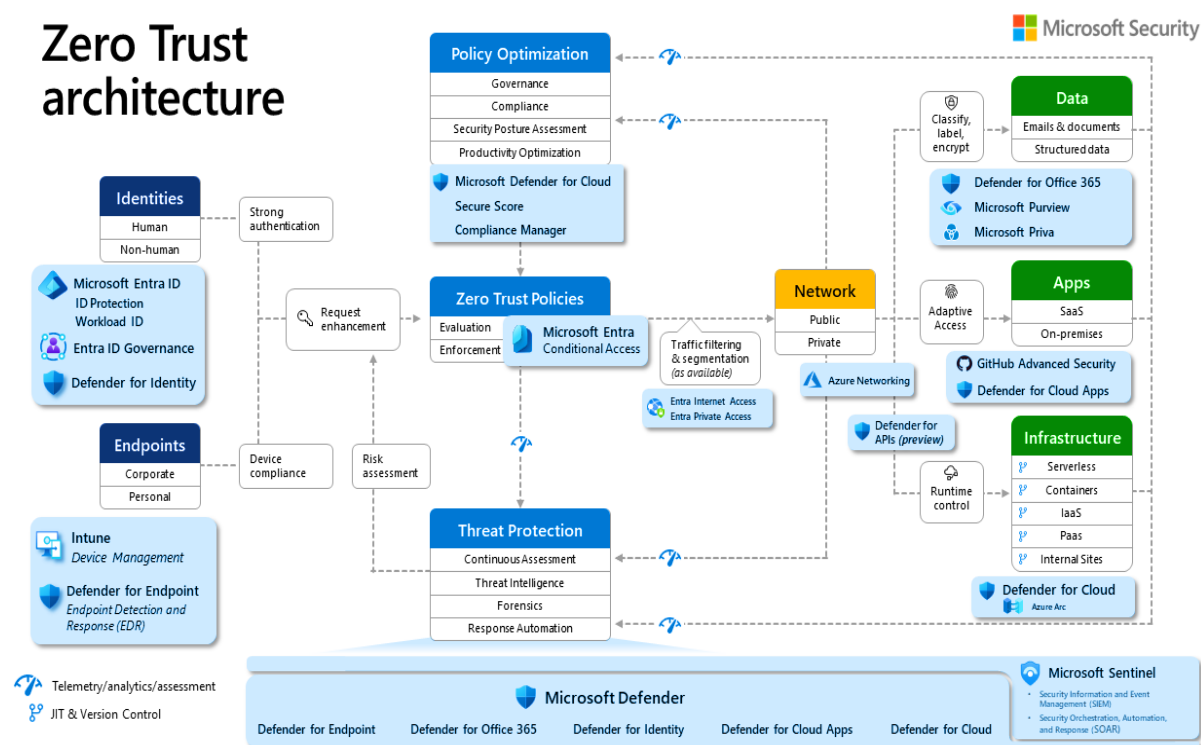


Рис. 4.8. Блок-схема архітектури Zero Trust

Джерело: URL: <https://www.netacad.com/courses/cybersecurity/>

На рис. 4.8 представлено блок-схему архітектури Zero Trust, яка демонструє ключові елементи моделі: користувачі, пристрої, додатки та дані під'єднані через систему перевірки доступу, що базується на політиках безпеки.

У центрі схеми розташовані компоненти прийняття рішень (Policy Decision Point) і виконання політик (Policy Enforcement Point), які забезпечують динамічний контроль кожного запиту. Показано, що навіть внутрішні користувачі не отримують автоматичного доступу, оскільки всі дії проходять перевірку ідентичності, контексту та ризиків у режимі реального часу. Такий підхід дозволяє мінімізувати латеральний рух атак у мережі, застосовувати мікросегментацію та забезпечувати багаторівневий захист корпоративних ресурсів. Це сучасна схема, яка відповідає підходам, описаним у рекомендаціях NIST SP 800-207 та практиці провідних виробників безпеки (Microsoft, Palo Alto, Cisco). Вона відображає актуальні тенденції інтеграції Zero Trust із хмарними сервісами, багатофакторною аутентифікацією та системами моніторингу поведінки користувачів.

Таким чином, еволюція моделей контролю доступу простежується від класичних і гнучких підходів (DAC) до суворих і формалізованих (MAC), далі до масштабованих і керованих ролями (RBAC), гнучких атрибутивних рішень (ABAC) та, зрештою, до концепції Zero Trust. Остання стала відповіддю на сучасний динамічний ландшафт загроз, забезпечуючи багаторівневий і контекстно-залежний захист корпоративних середовищ. Зараз на практиці застосовується багаторівневий підхід, що поєднує переваги різних моделей для досягнення комплексної безпеки. Це дозволяє підприємствам залишатися гнучкими, знижуючи ризики як внутрішніх, так і зовнішніх атак. У майбутньому контроль доступу дедалі більше інтегруватиметься з технологіями штучного інтелекту та автоматизованого реагування.

4.2. МІЖМЕРЕЖЕВІ ЕКРАНИ (СТАТИЧНІ, ДИНАМІЧНІ, NGFW) ТА ЇХ РОЛЬ У БАГАТОРІВНЕВОМУ ЗАХИСТІ

Міжмережеві екрани є ключовим елементом архітектури кіберзахисту, забезпечуючи фільтрацію та контроль мережевого трафіку відповідно до визначених політик безпеки. Вони формують перший бар'єр між внутрішніми ресурсами підприємства та зовнішніми загрозами, запобігаючи несанкціонованому доступу, витоку даних та розповсюдженню шкідливого програмного забезпечення. У сучасних умовах багаторівневого захисту міжмережеві екрани виступають основою для побудови захищеної периметрової інфраструктури та поєднуються з іншими інструментами безпеки (IDS/IPS, SIEM, антивірусні системи, механізми Zero Trust).

Сьогодні все частіше впроваджуються хмарні міжмережеві екрани (Cloud Firewalls), здатні масштабуватися під потреби підприємства та забезпечувати захист у розподілених інфраструктурах. NGFW інтегрують функції машинного

навчання для виявлення невідомих атак і застосовують технології Sandboxing для аналізу підозрілих об'єктів. З розвитком 5G та IoT-мереж особливого значення набуває підтримка високої пропускної здатності та захист величезної кількості підключених пристроїв. Крім того, сучасні міжмережеві екрани орієнтовані на глибоку інтеграцію з концепціями SASE (Secure Access Service Edge), поєднуючи мережеві й безпекові функції в єдиній платформі.

Статичні міжмережеві екрани

Статичні екрани (packet filtering firewalls) є найпростішим і найстарішим різновидом, що працює на мережевому та транспортному рівнях. Вони аналізують пакети даних за базовими параметрами — IP-адресою джерела і призначення, номером порту, протоколом — і приймають рішення «дозволити» або «заборонити» згідно з налаштованими правилами. Їхня головна перевага — висока швидкодія та простота у впровадженні. Проте вони не мають здатності відстежувати контекст сесії, тому вразливі до атак із підбійкою адрес (IP spoofing) чи до складніших загроз на прикладному рівні. Сьогодні статичні екрани застосовуються переважно як базовий рівень фільтрації або у поєднанні з більш складними механізмами.

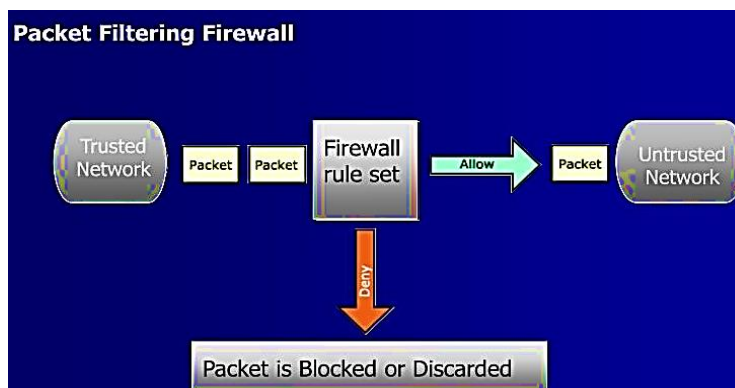


Рис. 4.9. Схема роботи пакетного фільтру (Packet Filtering Firewall)

Джерело: URL: <https://www.netacad.com/courses/cybersecurity/>

На схемі показано, як пакетний фільтр перевіряє окремі пакети мережевого трафіку, аналізуючи їхні заголовки на основі правил (IP-адреси джерела/призначення, порти, протокол) і приймаючи рішення – дозволити чи заблокувати. Пакет спочатку надходить з ненадійної мережі, проходить через блок із правилами фільтру, і лише якщо він відповідає умовам, переходить у захищену мережу. Якщо пакет не відповідає жодному правилу, він відкидається. Така схема ілюструє базовий рівень захисту та обмеженість підходу, адже фільтрація відбувається лише по заголовку, без врахування стану з'єднання чи контексту трафіку.

Статичні міжмережеві екрани працюють за принципом порівняння параметрів кожного пакета даних із попередньо визначеними правилами безпеки. Коли пакет проходить через екран, система аналізує його заголовок, перевіряючи IP-адресу джерела та призначення, номер порту та протокол, після чого приймає рішення — пропустити чи заблокувати його. Усі правила налаштовуються адміністратором вручну, що робить роботу екрана прозорою та передбачуваною, але водночас обмежує його здатність адаптуватися до складніших атак. Статичні екрани не враховують стан з'єднання й не відстежують послідовність взаємодії між вузлами, тому вони ефективні лише для базового контролю трафіку. Їхнє використання актуальне як первинний бар'єр безпеки або у зв'язі з динамічними екранами та системами виявлення вторгнень, де вони виконують функцію швидкого відсіювання небажаних або очевидно небезпечних пакетів.

Динамічні міжмережеві екрани

Динамічні екрани (stateful inspection firewalls) з'явилися як розвиток статичних, оскільки вони враховують контекст мережевої сесії. Такі системи відстежують стан з'єднань і формують таблицю сесій, що дозволяє перевіряти легітимність трафіку в реальному часі. Це робить їх значно ефективнішими у захисті від атак типу TCP SYN Flood чи несанкціонованих спроб встановлення підключення. Динамічні екрани забезпечують баланс між продуктивністю та безпекою, однак вони все ще обмежені у виявленні атак на рівні додатків, наприклад SQL Injection чи XSS. Сьогодні їх активно інтегрують із системами IDS/IPS, щоб компенсувати ці обмеження.

Динамічні міжмережеві екрани працюють за принципом відстеження стану кожного з'єднання, яке проходить через мережу, і створення спеціальної таблиці сесій, де фіксуються всі параметри активних підключень. Коли новий пакет надходить у систему, він перевіряється не лише за окремими атрибутами, як у статичних екранах, а й зіставляється з даними цієї таблиці, що дозволяє визначити, чи належить він до вже дозволеної та легітимної сесії. Таким чином динамічний екран здатний відрізнати законні пакети від підроблених і блокувати спроби встановлення несанкціонованих підключень. Це значно підвищує його ефективність у протидії мережевим атакам на транспортному рівні, зокрема DoS- і Flood-атакам.

У стандартному описі часто не вистачає підкреслення того, що динамічні екрани працюють за принципом Stateful Inspection, тобто виконують глибший аналіз пакетів у контексті всієї сесії, а не лише на основі правил фільтрації. Також варто зазначити, що вони підтримують timeout-механізми для завершення «завислих» сесій, що дозволяє уникати перевантаження. У сучасних реалізаціях динамічні екрани інтегрують функції Deep Packet Inspection (DPI) для обмеженого аналізу вмісту пакетів, а також взаємодіють із

системами IDS/IPS для виявлення складніших загроз. Водночас їхня слабка сторона полягає в обмеженій здатності розпізнавати атаки на прикладному рівні, через що вони розглядаються не як самодостатній засіб, а як частина багаторівневого захисту в поєднанні з NGFW чи іншими системами безпеки.

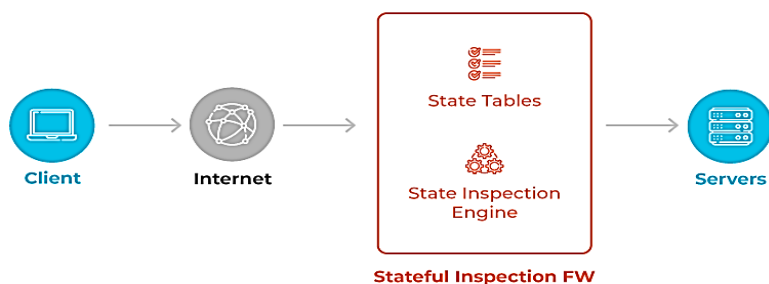


Рис. 4.10. Блок-схема динамічного (stateful) міжмережевого екрана
Джерело: URL: <https://www.netacad.com/courses/cybersecurity/>

Рис. 4.10 ілюструє, як динамічний міжмережевий екран відстежує стан з'єднань за допомогою модуля State Inspection Engine та State Tables. Пакети з мережі проходять через цей модуль, який перевіряє, чи відповідають вони встановленим з'єднанням у таблиці станів, і якщо так — пропускає їх далі до серверів. Якщо ж пакет не знайдений у таблиці чи не відповідає політикам, він аналізується або відкидається. Така схема показує основний механізм роботи stateful firewall — збереження контексту сесій і гнучкий контроль доступу в мережевому трафіку.

NGFW (Next-Generation Firewall)

Міжмережеві екрани нового покоління (NGFW) є еволюцією класичних підходів, поєднуючи можливості динамічної інспекції з розширеним аналізом на прикладному рівні. NGFW здатні ідентифікувати конкретні додатки (наприклад, Skype, WhatsApp, BitTorrent) незалежно від використовуваного порту чи протоколу. Вони включають вбудовані модулі для запобігання вторгненням (IPS), антивірусні механізми, систему контролю доступу користувачів і навіть підтримку технологій Sandboxing для перевірки підозрілих файлів. Додатково NGFW інтегруються з SIEM-системами, підтримують політики Zero Trust і використовують машинне навчання для адаптивного виявлення нових загроз. Основним недоліком NGFW є висока вартість і складність впровадження, але їхня ефективність у багаторівневому захисті робить їх стандартом для сучасних корпоративних мереж.

NGFW працюють за принципом багаторівневої перевірки трафіку, поєднуючи класичну фільтрацію пакетів, динамічний контроль стану з'єднань і глибоку інспекцію на рівні додатків. Коли пакет надходить до системи, він спочатку проходить базову перевірку відповідності політикам доступу, після

чого аналізується на предмет належності до конкретної програми чи сервісу, незалежно від порту чи протоколу. Далі задіюються механізми виявлення вторгнень (IPS), які здатні блокувати відомі сигнатури атак, а також поведінковий аналіз для виявлення аномалій у трафіку. NGFW також використовують технології Sandboxing, що дозволяють у віртуальному середовищі запускати підозрілі об'єкти та визначати їхню шкідливу активність до проникнення у внутрішню мережу.

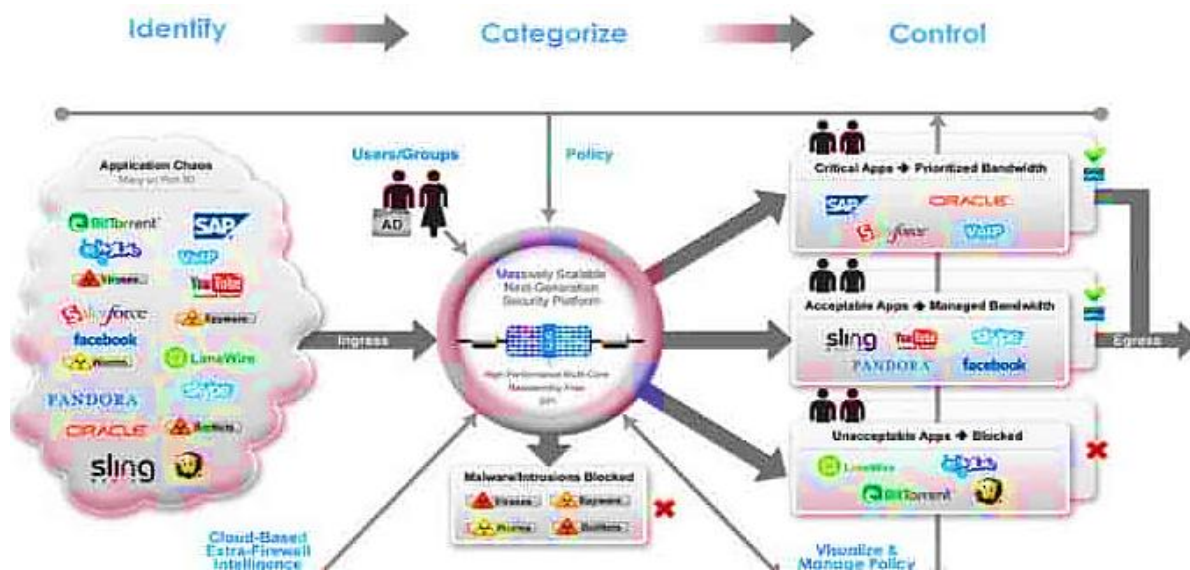


Рис. 4.11. Архітектура NGFW: класифікація та контроль додатків
Джерело: URL: <https://www.netacad.com/courses/cybersecurity/>

На цій схемі показано, як NGFW ідентифікує трафік не лише за IP/портом, але й за конкретними додатками або категоріями, застосовує гнучкі політики контролю, блокує небажані програми й фільтрує доступ залежно від груп користувачів. Вона також демонструє, як NGFW використовує модулі deep packet inspection (DPI), логування й механізми реагування на інциденти, щоб аналізувати зміст трафіку. Крім того, схема відображає, що рішення про дозвіл чи заборону приймаються на основі складних політик, а не простих правил фільтрації. Такий рисунок добре ілюструє сутність NGFW — комбінування традиційного фільтрування з аналізом на рівні додатків та адаптивним контролем доступу.

У стандартному описі часто не вистачає акценту на тому, що NGFW є не лише міжмережовим екраном, а цілісною платформою безпеки, яка поєднує функції IDS/IPS, антивірусу, URL-фільтрації, SSL/TLS-дефрагментації й навіть контролю політик для IoT-пристроїв. Вони підтримують інтеграцію з хмарними середовищами та забезпечують захист у мультихмарних інфраструктурах, де класичні екрани неефективні. Сучасні NGFW активно застосовують машинне

навчання для виявлення нульових днів (Zero-Day) і невідомих атак, що робить їх адаптивними до нових сценаріїв загроз. Крім того, важливою особливістю є підтримка централізованого управління та оркестрації, що дозволяє підприємствам уніфікувати політики безпеки в розподілених середовищах і забезпечувати узгоджений контроль на всіх рівнях мережевої архітектури.

Роль у багаторівневому захисті

Міжмережеві екрани — від простих статичних до сучасних NGFW — виконують роль фундаментального бар'єра, але їхня ефективність суттєво зростає у складі багаторівневої системи безпеки. У такій архітектурі статичні фільтри можуть відсікати найпростіший шкідливий трафік, динамічні — контролювати стан з'єднань і протидіяти базовим атакам, а NGFW — виявляти й блокувати складні багаторівневі загрози на рівні застосунків. Поєднання цих підходів із системами IDS/IPS, SIEM, аналітикою поведінки користувачів та концепцією Zero Trust створює адаптивний та стійкий до сучасних атак захисний контур. Саме тому міжмережеві екрани залишаються базовим, але водночас одним із найбільш критичних елементів комплексної кібербезпеки.

Роль міжмережевих екранів у багаторівневому захисті полягає у формуванні цілісного захисного периметра, де кожен рівень виконує власну функцію та компенсує обмеження інших. Статичні фільтри швидко й ефективно блокують очевидно небезпечний або небажаний трафік, динамічні екрани відстежують стан сесій і дозволяють протидіяти складнішим атакам на транспортному рівні, а NGFW забезпечують контроль на рівні додатків і застосовують глибокий аналіз пакетів. Усі ці рішення в комплексі взаємодіють із системами IDS/IPS, SIEM, а також із Zero Trust-архітектурою, що дозволяє будувати багатошаровий захист із можливістю оперативного реагування на загрози.

У типових викладах часто не вистачає кількох критично важливих аспектів — це підкреслення ролі хмарних і гібридних середовищ, де міжмережеві екрани виконують не лише периметрову, а й сегментуючу функцію, забезпечуючи поділ мережі на ізольовані зони. Важливо зазначити і інтеграцію з технологіями штучного інтелекту та машинного навчання, які дозволяють екранам прогнозувати аномалії й виявляти атаки нульового дня. Крім того, у сучасних сценаріях вони підтримують SASE (Secure Access Service Edge) та XDR-платформи, що забезпечує централізоване управління безпекою у розподілених інфраструктурах. Усе це робить міжмережеві екрани не просто засобом базового контролю трафіку, а критичним елементом адаптивної оборонної стратегії підприємства.

Таблиця 4.1. Порівняльна таблиця: роль статичних, динамічних і NGFW у багаторівневому захисті

Тип міжмережевого екрана	Рівень роботи	Переваги	Недоліки	Сучасні сфери застосування
Статичні (Packet Filtering)	Мережевий і транспортний рівні (IP-адреси, порти, протоколи)	- Простота реалізації та конфігурації - Висока швидкодія - Мінімальні ресурси для роботи	- Не враховують стан сесій - Вразливі до IP spoofing та складніших атак - Не аналізують прикладний рівень	- Базовий фільтр трафіку у малих мережах - Допоміжний рівень у багаторівневій архітектурі
Динамічні (Stateful Inspection)	Мережевий, транспортний рівні з урахуванням контексту сесії	- Відстеження стану з'єднань - Захист від TCP SYN Flood та несанкціонованих підключень - Баланс між продуктивністю і безпекою	- Обмежений аналіз прикладного рівня - Уразливі до складних атак (SQL Injection, XSS) - Вимагають більше ресурсів, ніж статичні	- Корпоративні мережі середнього рівня - Інтеграція з IDS/IPS для комплексного захисту
NGFW (Next-Generation Firewall)	Усі рівні, включно з прикладним (Application Layer)	- Ідентифікація додатків незалежно від порту - Вбудовані IPS, антивірус, URL-фільтрація - Підтримка Sandboxing і ML для виявлення Zero-Day атак - Інтеграція з SIEM та Zero Trust	- Висока вартість - Складність адміністрування - Вимогливість до ресурсів	- Великі підприємства та критична інфраструктура - Хмарні та мультимарні середовища - Підтримка SASE, XDR, IoT-захист

Таблиця показує, що статичні екрани підходять для базового рівня захисту, динамічні — для контекстної перевірки з'єднань, а NGFW забезпечують найповніший функціонал і є стандартом для сучасних корпоративних мереж. Важливо зазначити, що сучасні компанії дедалі частіше комбінують ці рішення, створюючи багаторівневі системи безпеки для зниження ризиків на різних рівнях мережі. NGFW у таких сценаріях інтегруються з SIEM і системами автоматизованого реагування, що дозволяє значно скоротити час виявлення та нейтралізації загроз. Динамічні екрани залишаються оптимальним варіантом для балансування між продуктивністю і безпекою в середніх мережах, тоді як статичні можуть застосовуватися як швидкий початковий фільтр. У майбутньому роль

NGFW ще більше зростатиме завдяки інтеграції з технологіями штучного інтелекту та Zero Trust-архітектурою.

4.3. СИСТЕМИ ВИЯВЛЕННЯ ТА ЗАПОБІГАННЯ ВТОРГНЕНЬ (IDS/IPS, HIDS, NIDS)

Системи виявлення та запобігання вторгнень (IDS/IPS) є ключовими елементами архітектури кіберзахисту, що забезпечують моніторинг, аналіз та блокування шкідливої активності в інформаційно-комунікаційних системах. Вони виконують роль «сенсорів безпеки», які ідентифікують підозрілу поведінку, фіксують спроби атак та забезпечують швидке реагування. У сучасних корпоративних мережах IDS/IPS інтегруються з SIEM-системами, платформами автоматизованого реагування (SOAR) та концепцією Zero Trust, створюючи багаторівневий захисний контур. Сьогодні все більше уваги приділяється інтеграції IDS/IPS із хмарними середовищами та сервісами, що дозволяє забезпечувати захист у мультихмарних і гібридних інфраструктурах. Для підвищення точності виявлення активно застосовуються алгоритми машинного навчання та штучного інтелекту, здатні розпізнавати Zero-Day атаки та невідомі загрози. Новітні рішення також підтримують аналіз зашифрованого трафіку (TLS/SSL inspection), що є критично важливим через зростання частки шифрованих даних у мережах. Крім того, IDS/IPS інтегруються з XDR-платформами, що дозволяє здійснювати кореляцію подій із різних джерел і формувати повний ланцюг атаки. У майбутньому очікується ще тісніше поєднання цих систем із технологіями автоматизованого реагування та поведінкової аналітики для мінімізації часу реагування на інциденти.

IDS (Intrusion Detection System)

IDS — це системи, що аналізують мережевий трафік або активність хостів з метою виявлення ознак атак. Вони можуть базуватися на сигнатурному підході (порівняння із відомими шаблонами атак) або на поведінковому (виявлення аномалій). IDS не блокують трафік напямую, а сповіщають адміністратора чи інші системи про інцидент. Сьогодні активно розвиваються методи застосування машинного навчання та штучного інтелекту для IDS, що дозволяє виявляти Zero-Day атаки та раніше невідомі загрози.

Рис. 4.12 ілюструє розташування систем IDS/IPS у мережевій інфраструктурі: мережевий пристрій підключений до проміського інтерфейсу, через який IDS прослуховує копії трафіку, а IPS розташовується безпосередньо «inline» у шляху проходження даних. У схемі видно, що IDS може працювати в режимі пасивного моніторингу (не втручаючись у потік) — просто сповіщаючи про підозрілі події, тоді як IPS здатна блокувати трафік у

режимі реального часу. Зображення також показує взаємодію з компонентами мережі (маршрутизаторами, брандмауєрами, серверами), що дає змогу зрозуміти, як системи IDS/IPS інтегруються в архітектуру охоронюваної мережі.

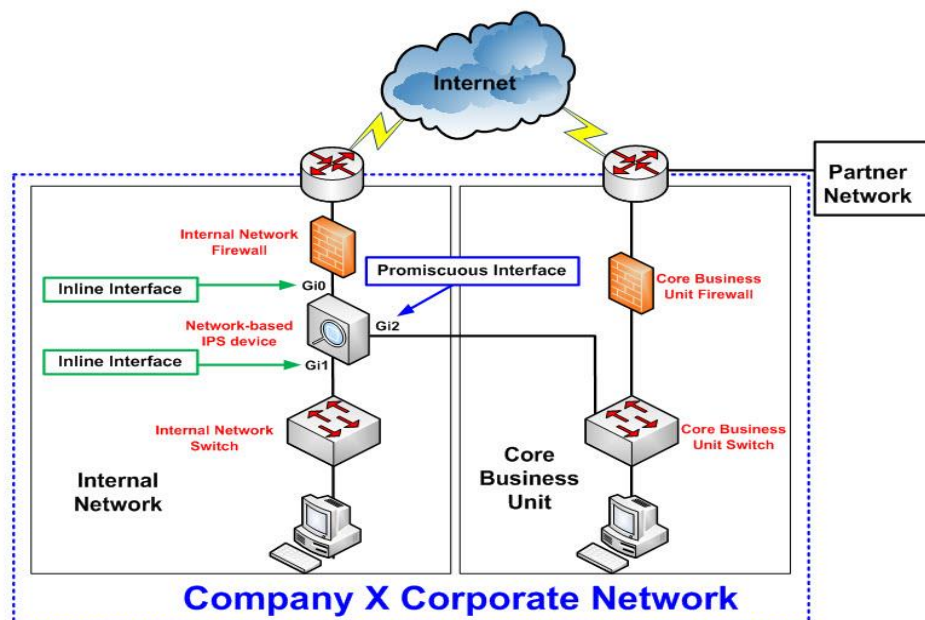


Рис. 4.12. Архітектура IDS/IPS у корпоративній мережі

Джерело: URL: <https://www.netacad.com/courses/cybersecurity/>

Схема на рис. 4.13 показує інтегровану систему IDS/IPS, що розміщена на віртуалізованих хостах (VM) через розподілені сенсори, які аналізують внутрішній трафік без відділення від мережевих сегментів. Події з різних сенсорів надходять на кореляційний рушій, який аналізує їх, зіставляє шаблони і виявляє складні ланцюги атак. У схемі видно, що система враховує не тільки мережевий трафік, а й внутрішньо віртуальні комунікації, що часто залишаються непоміченими традиційними IDS. Такий підхід підвищує здатність до виявлення латерального руху атак, атак між VM і внутрішніх загроз у дата-центрах чи хмарних середовищах.

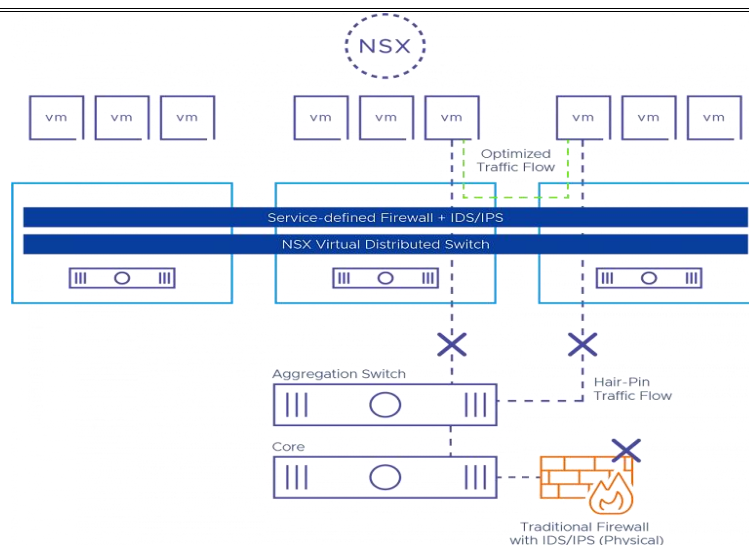


Рис. 4.13. Розподілена архітектура IDS/IPS з кореляцією подій у віртуалізованому середовищі

Джерело: URL: <https://www.netacad.com/courses/cybersecurity/>

IDS працює як «сенсор спостереження»: вона безперервно збирає події з мережі або хоста (пакети, логи ОС і додатків, системні виклики, зміни файлів), нормалізує їх і проганяє крізь набір детекторів — сигнатурних (пошук відомих шаблонів атак) і поведінкових (виявлення аномалій відносно базової «норми»). Результат кожного детектора зважується політиками кореляції: система зіставляє події між собою, підвищує або знижує «рейтинг підозрілості» та формує алерт із точною прив'язкою до об'єкта (IP/хост/користувач/процес), часу і контексту сесії. Далі сповіщення відправляються в SIEM/SOAR, де можуть автоматично запускати відповідні плейбуки (ізоляція вузла EDR-засобами, блокування IP на брандмауері, збір форензіки). У мережевому варіанті (NIDS) аналіз виконується пасивно на дзеркалі портів або TAP, а в хостовому (HIDS) — локальними агентами; обидва підходи часто поєднують для повного покриття ланцюга атаки.

У типових викладах про IDS часто не вистачає кількох критично важливих аспектів. По-перше, бракує акценту на керуванні помилковими спрацьовуваннями: потрібні системний тюнінг правил, винятки, придушення шуму та коректна базова лінія поведінки. По-друге, мало уваги приділяють видимості у зашифрованому трафіку — без TLS-декрипції або аналізу метаданих (зокрема JA3/JA3S-відбитків) можливості NIDS різко зменшуються. По-третє, часто оминають вимоги до стійкості проти обходів, таких як фрагментація, поліморфні пейлоади й протокольні аномалії, що потребує нормалізації трафіку та DPI. По-четверте, не розкривають питання продуктивності й масштабування: розподілені сенсори, буферизація, горизонтальне масштабування та зберігання довгострокових PCAP/логів для

форензика. І, нарешті, рідко визначають метрики ефективності (MTTD/MTTR, precision/recall для ML-моделей) і життєвий цикл правил/моделей — від режиму «alert-only» до поступового розширення покриття з контрольованим ризиком.

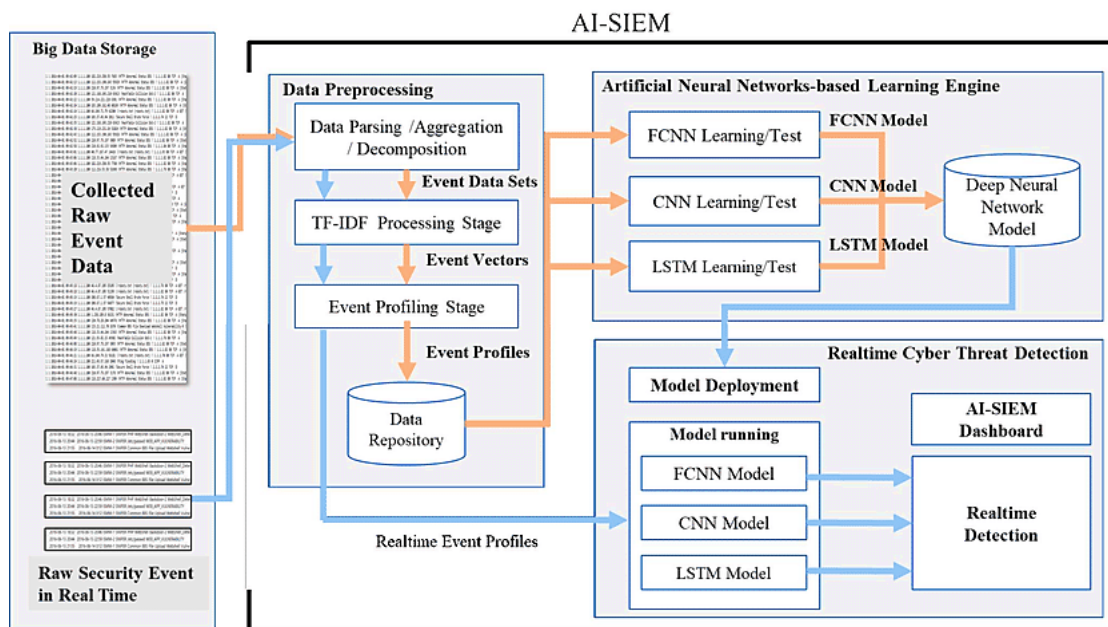


Рис. 4.14. Архітектура AI-SIEM з кореляцією подій у реальному часі
Джерело: URL: <https://www.netacad.com/courses/cybersecurity/>

Схема на рис. 4.14 демонструє, як AI/ML-моделі інтегруються в SIEM-систему для аналізу та класифікації подій, які надходять із різних джерел (мережі, сервери, додатки). Вона показує етапи попередньої обробки даних (парсинг, агрегація, профілювання), побудови векторів подій та подальший запуск ML/нейронних мереж (CNN, LSTM та ін.) для класифікації або виявлення аномалій. Кореляційний рушій у схемі поєднує результати моделей із логами та іншими подіями для виявлення складних ланцюгів атак, створюючи більш контекстуальні алерти. Така архітектура дозволяє автоматично визначати «нормальні» поведінкові шаблони, пом'якшувати фальшиві спрацьовування й активніше виявляти складні зловмисні кампанії.

IPS (Intrusion Prevention System)

IPS — це розвиток IDS, що не лише виявляє, а й блокує шкідливу активність у режимі реального часу. Система може автоматично відкидати пакети, закривати з'єднання чи змінювати правила фільтрації для запобігання подальшому розвитку атаки. IPS працюють на базі сигнатурного аналізу, евристики та глибокої інспекції пакетів (DPI), а також підтримують інтеграцію з NGFW для комплексного контролю. Сучасні IPS здатні працювати у високонавантажених середовищах, включаючи хмарні й мультихмарні

інфраструктури, забезпечуючи захист від DDoS, ботнетів, експлойтів і складних багаторівневих атак.

IPS працює «inline» у трафіковому шляху й аналізує кожен пакет та потік у контексті всієї сесії: спершу виконується нормалізація (дефрагментація, реконструкція TCP-потоків, усунення неоднозначностей протоколів), далі — багаторівнева перевірка політик, сигнатур і поведінкових правил. Механізм DPI розпізнає протоколи й додатки незалежно від портів, декодує їхні поля, зіставляє з базами відомих експлойтів/IOC та запускає евристику для виявлення підозрілих послідовностей, темпу трафіку чи аномальних станів. Якщо подія класифікується як шкідлива або високоризикова, IPS миттєво застосовує дію: дроп/резет з'єднання, тимчасове блокування джерела, rate-limit або динамічне оновлення ACL/правил NGFW; паралельно створює інцидент і надсилає телеметрію в SIEM/XDR. Для зниження шуму використовується кореляція з контекстом (ідентичність користувача, стан хоста/EDR, гео/час, критичність активу) і механізми довіри до джерел (allow-list, reputation feeds), а також «graceful degradation» під навантаженням — пріоритизація критичних сигнатур і відкладений глибинний аналіз.

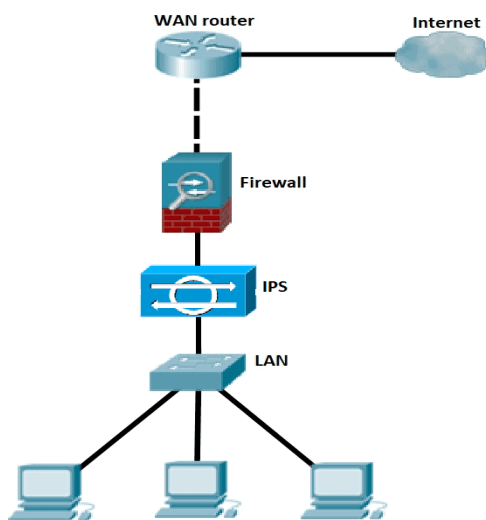


Рис. 4.15. Архітектура пристрою IPS у мережі

Джерело: URL: <https://www.netacad.com/courses/cybersecurity/>

Схема на рис. 4.15 показує, як IPS розміщується inline після брандмауера, безпосередньо у шляху трафіку, щоб мати змогу блокувати атаки в режимі реального часу. У ній видно, що трафік із Інтернету проходить через firewall, потім через IPS, а вже потім — до внутрішньої мережі (LAN). IPS аналізує пакети, перевіряє їх через бази сигнатур та правила і проводить автоматичне блокування або приймає інші заходи. Такий рисунок ілюструє типове інтерактивне розміщення IPS — між зовнішнім та внутрішнім сегментами мережі — для активного захисту трафіку.

На схемі рис. 4.16 показано, як сенсори IDS/IPS розташовані в різних сегментах мережі й передають події до централізованого кореляційного рушія. AI/ML-модулі аналізують дані подій у реальному часі, виділяють аномалії та класифікують інциденти за ризиком. Кореляційний рушій поєднує інформацію від декількох сенсорів і контекст (наприклад, часові паттерни, поведінка користувачів, загрози від відомих джерел). Далі приймається рішення про блокування чи подальше дослідження, з відправкою результатів до SIEM або систем реакції (SOAR).

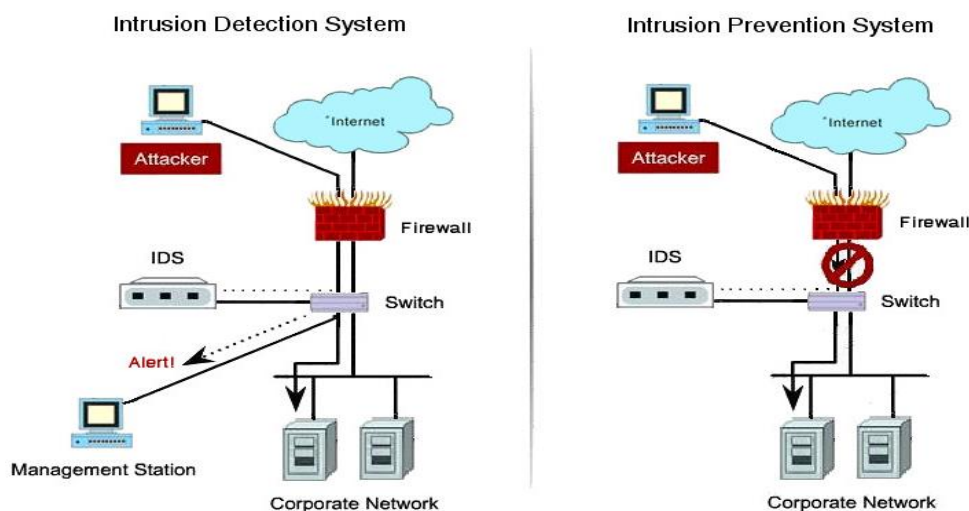


Рис. 4.16. Інтегрована архітектура IDS/IPS із кореляцією подій
Джерело: URL: <https://www.netacad.com/courses/cybersecurity/>

У типових викладах про IPS бракує: по-перше, уваги до TLS/SSH-видимості — без розшифрування або аналізу телеметрії/метаданих ефективність суттєво падає; потрібні проксі/кільцеві агенти або selective decryption з урахуванням приватності. По-друге, детальної стратегії тюнінгу й тестування у “prevention”-режимі: спершу alert-only у тіньовому режимі, A/B-включення правил, контроль FP/FN та поетапний перехід до блокування. По-третє, плану стійкості та обходів: захист від евазивних технік (фрагментація, обфускація, protocol tunneling), регулярне оновлення декодерів протоколів і ввімкнена нормалізація. По-четверте, продуктивності й безвідмовності: апаратні прискорювачі (ASIC/DPDK), кластеризація active/active, bypass-карти на випадок збою, а також QoS/політики, щоб IPS не став «вузьким місцем». І нарешті, операційної інтеграції: чіткі плейбуки SOAR, зворотний зв'язок із EDR/NGFW, ведення показників MTTD/MTTR і процесу керування життєвим циклом правил та моделей (versioning, rollback, peer-review).

HIDS (Host-based Intrusion Detection System)

HIDS орієнтовані на моніторинг і аналіз подій конкретного хоста або сервера. Вони контролюють системні журнали, зміни у файлах, процесах та конфігураціях, а також відстежують спроби ескалації привілеїв. HIDS є надзвичайно ефективними для виявлення внутрішніх загроз та інсайдерських атак, які не завжди можна помітити через мережевий трафік. Сьогодні HIDS інтегруються з EDR-рішеннями (Endpoint Detection and Response), що дозволяє здійснювати глибший аналіз на рівні кінцевих точок та швидко реагувати на інциденти.

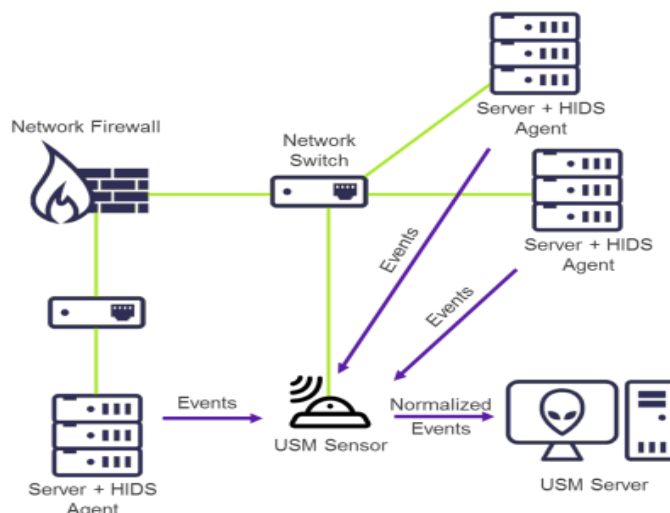


Рис. 4.17. Архітектура HIDS з агент-сенсором і центральним кореляційним сервером

Джерело: URL: <https://www.netacad.com/courses/cybersecurity/>

Схема на рис. 4.17 ілюструє HIDS-агентів, встановлених на різних серверах, які збирають події (журнали, зміни файлів, інші події системи) і передають їх на центральний сенсор (USM Sensor). Центральний сенсор нормалізує події, корелює їх і передає в систему управління та аналізу (USM Server) для подальшого аналізу й дій. Вона також демонструє, що HIDS може працювати спільно з мережею та брандмауерами, щоб отримувати контекст зовнішніх подій. Такий рисунок добре ілюструє агент-серверну архітектуру HIDS у сучасному середовищі із централізованим корелюванням подій.

HIDS працюють за принципом постійного збору й аналізу даних із операційної системи та прикладних процесів, створюючи базову лінію нормальної активності хоста. При відхиленні від цієї базової лінії система фіксує потенційну загрозу та формує повідомлення для адміністратора або автоматизованої системи реагування. Вона здатна виявляти спроби модифікації критичних файлів, створення підозрілих облікових записів, несанкціоноване втручання у конфігураційні параметри чи виконання шкідливих процесів. Сучасні HIDS підтримують застосування машинного навчання для точнішої

класифікації поведінки, знижуючи кількість хибнопозитивних спрацювань. Крім того, вони можуть інтегруватися з SIEM та SOAR-платформами, забезпечуючи централізовану кореляцію подій та автоматизоване реагування. В умовах розподілених хмарних і гібридних інфраструктур NIDS дедалі частіше поєднують із XDR-рішеннями, створюючи єдиний контур безпеки, орієнтований як на превентивний захист, так і на швидку локалізацію загроз.

NIDS (Network-based Intrusion Detection System)

NIDS розташовуються у ключових сегментах мережі й аналізують трафік у реальному часі. Вони дозволяють виявляти зовнішні атаки, аномалії у взаємодії між вузлами та підозрілі потоки даних. Сучасні NIDS підтримують DPI, можуть розпізнавати протоколи на рівні додатків і виявляти атаки на веб-сервіси чи бази даних. У великих корпоративних середовищах NIDS поєднуються з SIEM та XDR-платформами, що дає змогу будувати аналітику на основі кореляції подій і формувати повний ланцюг атаки.

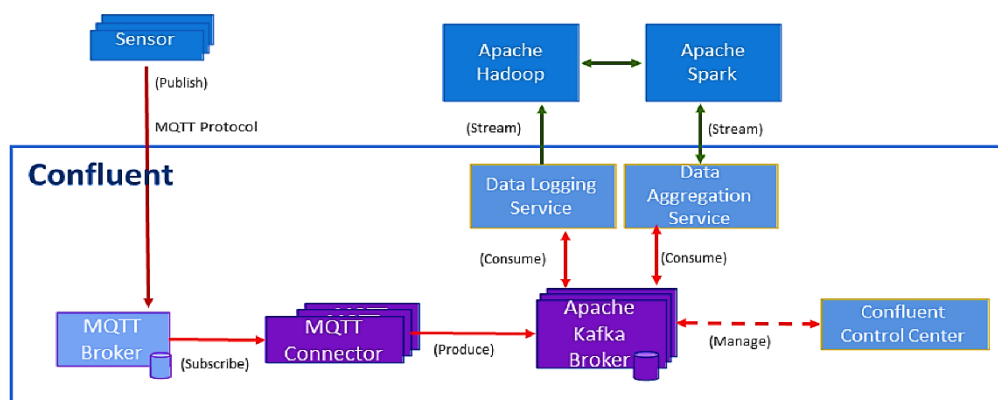


Рис. 4.18. Хмарна архітектура NIDS із обробкою великих даних і розподіленими сенсорами

Джерело: URL: <https://www.netacad.com/courses/cybersecurity/>

Схема на рис. 4.18 демонструє розподілені NIDS-сенсори, розміщені у хмарній інфраструктурі, які збираючи трафік, передають події у централізовану систему агрегації через Kafka або інші брокери повідомлень. У центрі зображено модуль «Data Logging / Aggregation», який збирає та нормалізує дані з усіх сенсорів для подальшої обробки. Далі дані надходять до аналітичних компонентів (Apache Spark, Hadoop), які виконують кореляцію, класифікацію та AI/ML-аналіз подій. Таким чином, ця архітектура підкреслює сучасний підхід до масштабованого NIDS, орієнтованого на великі обсяги даних і використання технологій Big Data для глибшого аналізу загроз.

NIDS працює пасивно в ключових точках мережі (SPAN/port-mirror, мережеві TAP), копіюючи трафік і аналізуючи його в режимі реального часу без впливу на потік даних. Система нормалізує пакети (дефрагментація IP,

реконструкція TCP-сесій), виконує глибоку інспекцію (DPI) та декодує протоколи прикладного рівня, після чого застосовує комбінацію сигнатурних правил і поведінкових моделей для виявлення відомих атак і аномалій (сканування, експлойти, lateral movement). Події корелюються у часовому та сесійно-контекстному вимірах (клієнт ↔ сервер, користувач/хост, гео/час), присвоюється пріоритет ризику, а результати надсилаються до SIEM/XDR для подальшого збагачення (репутація IP/домена, дані з EDR/брандмауерів) і запуску плейбуків реагування. Щоб підвищити видимість і точність, сучасні NIDS підтримують аналіз метаданих зашифрованих сесій (JA3/JA3S-відбитки, SNI, статистика потоків), масштабуються горизонтально за рахунок розподілених сенсорів і балансування навантаження, а також використовують ML-моделі для зниження хибнопозитивних спрацювань у високошвидкісних сегментах мережі.

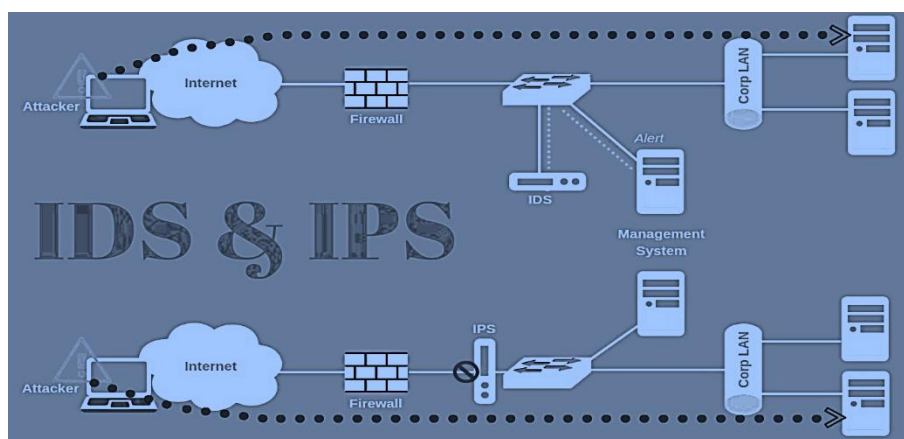


Рис. 4.19. Архітектура IPS/IDS із мережесим розміщенням
Джерело: URL: <https://www.netacad.com/courses/cybersecurity/>

На рис. 4.19 показано, як IPS розташовується inline у шляху мережевого трафіку — після брандмауера — щоб мати змогу блокувати шкідливі пакети в режимі реального часу, тоді як IDS працює пасивно на копії трафіку і лише надсилає алерти. Видно, як події, зафіксовані сенсорами, передаються до системи управління/моніторингу для кореляції й реагування. Схема також підкреслює відокремлення ролей — один компонент відповідає за превенцію (IPS), інший — за детекцію (IDS). Такий тип графіки добре ілюструє сучасну архітектуру систем виявлення та запобігання вторгнень у корпоративній мережі.

Сучасні тенденції розвитку IDS/IPS

Сучасні тенденції розвитку IDS/IPS визначають їхнє перетворення з окремих сенсорів безпеки на комплексні аналітичні платформи. Інтеграція з технологіями штучного інтелекту та машинного навчання дозволяє системам

виявляти Zero-Day атаки й аномальну поведінку навіть без попередньо визначених сигнатур, що суттєво підвищує їхню ефективність у динамічному середовищі кіберзагроз.

Хмарні рішення IDS/IPS стають невід'ємною частиною архітектури SASE, забезпечуючи моніторинг у розподілених інфраструктурах, мультихмарних середовищах та на віддалених робочих місцях. Автоматизація реагування завдяки тісній інтеграції з SOAR дозволяє скоротити час між виявленням інциденту та його блокуванням, зменшуючи ризики для бізнес-процесів. Глибокий аналіз шифрованого трафіку за допомогою TLS/SSL inspection дає змогу виявляти загрози у зашифрованих потоках, які раніше залишалися невидимими для класичних систем. Крім того, IDS/IPS інтегруються в концепцію Zero Trust, виконуючи роль постійних сенсорів моніторингу та контролю довіри в реальному часі, що є критично важливим для сучасних корпоративних середовищ. У найближчій перспективі очікується ще ширше використання розподілених сенсорів, аналізу великих даних (Big Data) та XDR-платформ, які дозволяють створювати цілісну картину атак і формувати передиктивну аналітику для проактивного захисту.

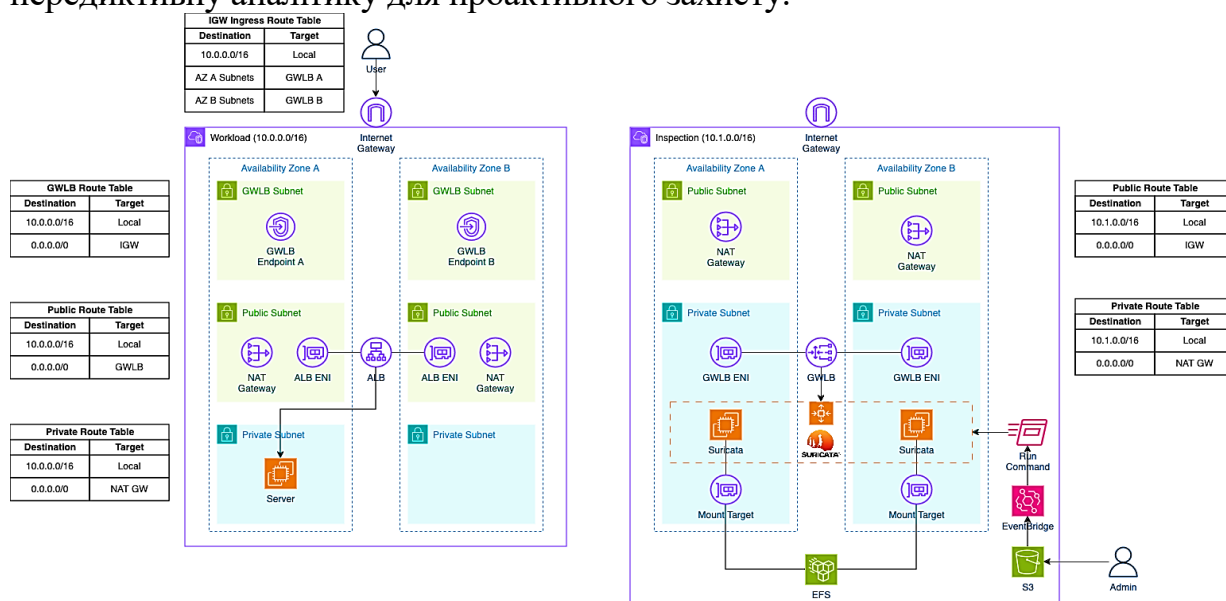


Рис. 4.20. Масштабована хмарна архітектура IDS/IPS на базі Suricata в AWS

Джерело: URL: <https://www.netacad.com/courses/cybersecurity/>

Схема на рис. 4.20 демонструє, як сенсори IDS/IPS (в даному випадку Suricata) розгортаються в хмарному середовищі за допомогою AWS Gateway Load Balancer для балансування трафіку й масштабування відповідно до навантаження. Вона показує потік інгрес- та егрес-трафіку через інспекційний вузол, що аналізує дані, реалізуючи сигнатурні та поведінкові алгоритми, і

здатен реагувати на загрози. У схемі також видно інтеграцію з автомасштабуванням (Auto Scaling) та централізованим сховищем логів і правил (EFS, S3). Це сучасний приклад реалізації IDS/IPS як частини хмарної інфраструктури, придатний для мультихмарних, гібридних та розподілених середовищ.

Системи IDS/IPS, включаючи HIDS та NIDS, залишаються фундаментально важливими складовими багаторівневого захисту. Вони не лише дозволяють виявляти та блокувати відомі загрози, а й формують основу для сучасних підходів до кібербезпеки, що базуються на аналітиці поведінки, автоматизації та штучному інтелекті. Сьогодні їхнє значення зростає у зв'язку з поширенням хмарних обчислень, гібридних мереж та віддаленої роботи, де IDS/IPS виступають як центральні сенсори безпеки для збереження цілісності корпоративних систем.

4.4. ІНСТРУМЕНТИ SIEM ТА SOAR ДЛЯ ЦЕНТРАЛІЗОВАНОГО МОНІТОРИНГУ І РЕАГУВАННЯ

Інструменти SIEM та SOAR для централізованого моніторингу і реагування — це комплекс рішень, де SIEM (Security Information and Event Management) збирає, нормалізує та корелює події безпеки з усіх джерел (мережа, хости, хмари, додатки) і в реальному часі виявляє інциденти, а SOAR (Security Orchestration, Automation and Response) оркеструє інструменти та автоматизує реагування за сценаріями (playbooks): блокує IP, ізолює вузли, відключає акаунти тощо. Разом вони створюють «мозок і руки» SOC, скорочуючи MTTD/MTTR, зменшуючи ручну рутину та підвищуючи якість рішень завдяки кореляції контексту. Сучасні реалізації підтримують AI/ML-аналітику, інтеграцію з XDR, роботу в мультихмарних середовищах і узгоджуються з принципами Zero Trust.

Інтеграція SIEM та SOAR забезпечує безперервний цикл управління інцидентами: від збору телеметрії до автоматизованого реагування й постінцидентного аналізу. Це дозволяє підприємствам оперативно реагувати навіть на складні багатовекторні атаки, мінімізуючи ризик ескалації. Завдяки гнучким сценаріям SOAR можливо стандартизувати процедури реагування та забезпечити відповідність регуляторним вимогам. SIEM-системи часто стають ядром централізованого моніторингу, тоді як SOAR виступає рушієм автоматизації процесів і масштабованості реагування. У поєднанні вони підтримують концепцію «cyber resilience», підвищуючи стійкість підприємства до кіберзагроз. Таким чином, SIEM та SOAR є критично важливими складовими сучасних систем захисту інформації, що формують основу ефективного SOC.

На схемі (рис. 4.21) показано, як події з різноманітних джерел (мережевий трафік, логи брандмауерів, проксі, додатків, баз даних) надходять у систему через канал подій (наприклад, через Apache Kafka або інші потокові сервіси). Потім події нормалізуються, фільтруються та корелюються всередині SIEM-модуля, а сигнали передаються до SOAR для автоматизованого реагування за попередньо визначеними сценаріями (playbooks). Схема також демонструє, що SOAR може керувати діями в системах EDR, брандмауерах, антивірусах чи інших компонентах інфраструктури. Така архітектура добре відображає сучасний підхід до побудови інтегрованої екосистеми безпеки, де SIEM забезпечує аналітику, а SOAR — автоматизацію реагування.

Build A Real-time SIEM / SOAR Pipeline

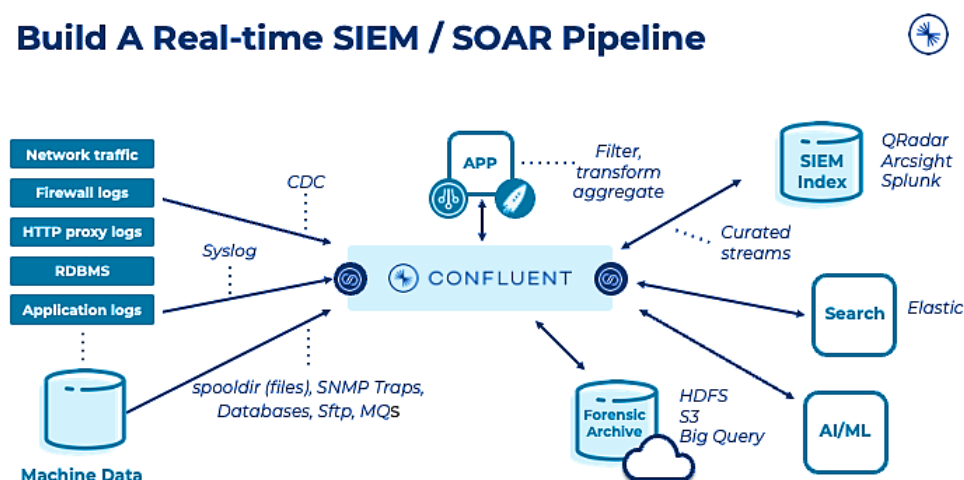


Рис. 4.21. Архітектура SIEM/SOAR з аналізом подій у реальному часі
Джерело: URL: <https://www.netacad.com/courses/cybersecurity/>

Принцип дії архітектури, зображеної на схемі (рис. 4.21), можна описати поетапно. По-перше, усі події безпеки надходять із різноманітних джерел — мережевого обладнання, серверів, прикладного ПЗ, хмарних платформ чи баз даних. Щоб уніфікувати їхній формат і зробити придатними для подальшої обробки, дані потрапляють у канал подій (event bus), яким часто виступають високопродуктивні брокери повідомлень на кшталт Apache Kafka, RabbitMQ чи Fluentd.

На другому етапі SIEM здійснює нормалізацію даних (переведення до єдиного формату), їх фільтрацію від шуму та кореляцію. Кореляційні правила дозволяють виявляти складні інциденти, що складаються з багатьох розрізнених подій, наприклад послідовність невдалих логінів і спроб ескалації привілеїв. Саме тут формується сигнал (alert), який позначає потенційну загрозу.

Далі відбувається передача сигналів до SOAR-системи. Вона активує заздалегідь підготовлені сценарії реагування (playbooks), які описують

покрокові дії: блокування IP-адреси на міжмережевому екрані, ізоляція скомпрометованого хоста в EDR, примусове завершення сесії користувача або зміна його облікових даних у каталозі.

SOAR взаємодіє з різними компонентами інфраструктури через API або агентів, виступаючи «центром управління», що оркеструє роботу інших засобів захисту. Це мінімізує ручні дії аналітиків SOC і дозволяє масштабувати реагування на сотні чи тисячі інцидентів.

Завдяки зворотному зв'язку результати реагування повертаються в SIEM, що збагачує подальший аналіз і дозволяє адаптувати кореляційні правила та сценарії. Таким чином, формується замкнений цикл: збір → аналітика → реагування → оптимізація, який забезпечує безперервне вдосконалення екосистеми кіберзахисту.

Інструменти SIEM та SOAR сьогодні дедалі частіше інтегруються в єдині платформи, що дозволяє не лише централізувати збір даних, а й повністю автоматизувати реагування. Завдяки цьому компанії отримують можливість створювати сучасні SOC із високим рівнем автономності та прогнозної аналітики. Поєднання SIEM з SOAR забезпечує безперервний моніторинг інфраструктури, миттєве реагування на загрози та скорочення впливу людського фактору. У перспективі ці системи дедалі ширше застосовуватимуть передиктивні алгоритми та штучний інтелект для формування автоматизованих рішень у режимі реального часу.

SIEM (Security Information and Event Management)

Системи SIEM є основою централізованого моніторингу інформаційної безпеки. Вони поєднують функції збору логів, нормалізації, кореляції подій та їх візуалізації у єдиній консолі. Сучасні SIEM-рішення (Splunk, IBM QRadar, Elastic SIEM, Microsoft Sentinel) працюють у реальному часі, використовуючи алгоритми машинного навчання для виявлення аномалій та Zero-Day атак. Вони здатні обробляти величезні обсяги даних з різних джерел — серверів, мережевих пристроїв, систем доступу, хмарних платформ і навіть IoT-пристроїв. Завдяки інтеграції з Big Data-аналітикою SIEM створюють цілісну картину подій безпеки, допомагаючи адміністраторам не лише реагувати на інциденти, а й виявляти приховані тенденції та загрози. Сучасні підходи передбачають інтеграцію SIEM із Zero Trust-архітектурою, що дає змогу забезпечити безперервний моніторинг довіри та виявлення внутрішніх загроз.

Архітектура SIEM

Системи SIEM побудовані на багаторівневій архітектурі, яка включає кілька ключових компонентів. На першому рівні працюють агенти збору даних, що інтегруються із серверами, мережевим обладнанням, хмарними сервісами та кінцевими точками для отримання логів і подій безпеки. Далі дані проходять через модуль нормалізації, який уніфікує різні формати та додає контекст — від

ідентифікації користувачів і геолокації до репутації IP-адрес. Важливим елементом є сховище логів або Data Lake, де інформація зберігається для пошуку та довгострокового аналізу. На рівні аналітичного двигуна працюють механізми кореляції та ML/AI-моделі, які виявляють аномалії, Zero-Day атаки та комплексні загрози. Усе це відображається в консолі SOC у вигляді дашбордів і звітів, забезпечуючи єдиний центр управління подіями безпеки.

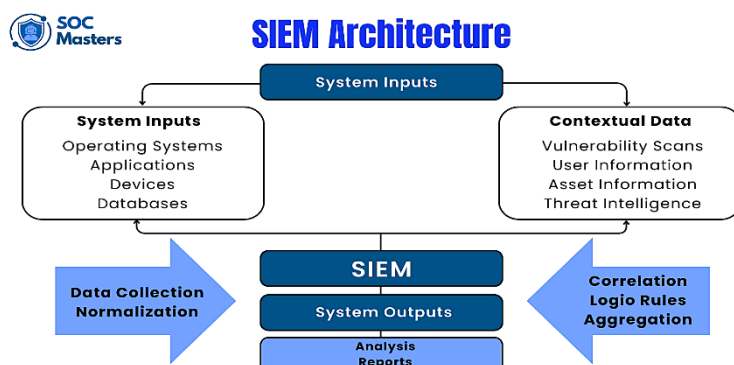


Рис. 4.22. Архітектура SIEM: початкова обробка, кореляція, аналітика
Джерело: URL: <https://www.netacad.com/courses/cybersecurity/>

На схемі показано, як події з різноманітних джерел (сервери, пристрої, хмари, додатки) входять у систему через входні модулі (System Inputs), потім поділяються на «подієві дані» та «контекстуальні дані», які проходять нормалізацію та збагачення. Далі кореляційна логіка і правила агрегують ці події, зіставляючи їх у інциденти, що відправляються як вихідна інформація (System Outputs) — алерти, звіти, аналітика. Таким чином, рисунок демонструє ключові етапи архітектури SIEM: збір, нормалізацію, кореляцію та генерування аналітичних результатів, які потім представляються аналітикам SOC.

Принцип роботи SIEM

Робота SIEM відбувається поетапно, забезпечуючи повний цикл моніторингу та аналізу подій. Спершу система здійснює збір інформації з усіх джерел інфраструктури, після чого відбувається нормалізація та збагачення даних додатковим контекстом. Далі SIEM виконує кореляцію подій, поєднуючи їх у цілісні сценарії атак: наприклад, численні невдалі входи з різних IP перетворюються на єдиний інцидент brute force. На основі сигнатур, поведінкових профілів та алгоритмів ML/AI система виявляє загрози, після чого генерує алерти та передає їх адміністраторам чи в SOAR для автоматизованого реагування. Завершальний етап — аналіз і створення звітності для розслідувань, аудиту та відповідності міжнародним стандартам безпеки. Сучасні SIEM здатні працювати у гібридних і мультихмарних середовищах, підтримують інтеграцію із Zero Trust і XDR, що робить їх центральним елементом сучасних центрів безпеки SOC.

Налаштування та оптимізація SIEM

Правильне налаштування SIEM-системи є критичним фактором її ефективності. На етапі впровадження першочергово визначаються джерела подій: мережеве обладнання, сервери, бази даних, веб-додатки, системи автентифікації, хмарні сервіси. Для кожного джерела необхідно активувати логування безпеки та встановити агенти збору даних або налаштувати відправку через протоколи Syslog, SNMP чи API.

Далі формується базовий набір кореляційних правил, який відповідає стандартним сценаріям атак (brute force, lateral movement, privilege escalation). Важливо провести адаптацію правил до конкретної інфраструктури, щоб уникнути великої кількості хибнопозитивних спрацювань.

Наступним кроком є створення use cases (сценаріїв виявлення), які відображають найбільш імовірні та критичні для підприємства загрози. Для цього застосовуються як вбудовані шаблони SIEM, так і власні кастомні правила, що враховують специфіку бізнес-процесів.

Ключову роль відіграє налаштування порогів сповіщень: занадто низькі значення створюють «шум» і перевантажують SOC-аналітиків, а надто високі — можуть призвести до пропуску реальних інцидентів.

Для підвищення точності доцільно використовувати інтеграцію з Threat Intelligence платформами, які забезпечують актуальні дані про відомі шкідливі IP-адреси, домени, хеші файлів. Це дозволяє збагачувати події зовнішнім контекстом і швидше ідентифікувати атаки.

Завершальним етапом є оптимізація та регулярний аудит конфігурацій: перегляд правил, відсів малоефективних кореляцій, впровадження ML/AI-моделей для виявлення невідомих загроз, а також навчання SOC-аналітиків роботі з дашбордами і звітністю.

SOAR (Security Orchestration, Automation and Response)

SOAR-системи є наступним кроком у розвитку інструментів кіберзахисту, адже вони не тільки збирають і аналізують дані, але й автоматизують реагування на інциденти. Вони дозволяють інтегрувати різні рішення безпеки (SIEM, IDS/IPS, NGFW, антивірусні системи, хмарні сервіси) у єдину платформу. Основна функція SOAR — зменшення часу від виявлення до реагування завдяки автоматизації рутинних процесів: блокування IP-адреси, відключення компрометованого облікового запису, ізоляції зараженого вузла. Сучасні SOAR-рішення (Palo Alto Cortex XSOAR, Splunk SOAR, IBM Resilient, ServiceNow SecOps) активно застосовують AI/ML для динамічного визначення пріоритетності інцидентів та прогнозування сценаріїв атак. Крім того, вони

надають можливість створення «playbooks» — автоматизованих сценаріїв дій, які зменшують вплив людського фактора та стандартизують процес реагування.



Рис. 4.23. Ключові компоненти SOAR: оркестрація, реагування, автоматизація

Джерело: URL: <https://www.netacad.com/courses/cybersecurity/>

На рис. 4.23 представлено взаємодію основних компонентів SOAR — Orchestration, Automation та Response — у контексті життєвого циклу інцидентів. Він показує, що SOAR узгоджує різні інструменти й джерела даних (Threat Intelligence, SIEM, сенсори) у єдину систему, автоматизує рутинні завдання через плейбуки та забезпечує стандартизоване реагування. У схемі також видно, що SOAR підтримує процеси triage, enrichment та синтезу рішень, а дії виконуються через інтегровані системи (EDR, NGFW, IAM тощо). Така схема добре ілюструє архітектурний підхід SOAR як мозок, який поєднує розрізнені компоненти систем безпеки в єдину автоматизовану екосистему реагування.

Архітектура SOAR (Security Orchestration, Automation and Response)

Сучасна SOAR-платформа будується як модульна, API-орієнтована система з широким набором конекторів до джерел подій і засобів реагування (SIEM, EDR/XDR, IDS/IPS, NGFW, проксі, DLP, CASB, хмарні платформи, системи керування ідентичністю, поштові шлюзи, тикетинг/ITSM). На вході працює шар інтеграцій (конектори, вебхуки, черги подій), що приймає інциденти та артефакти (IOC, файли, URL, хеші, PCAP, лог-фрагменти). Далі шар нормалізації та збагачення уніфікує формати, виконує дедуплікацію/кластеризацію, збагачує контекстом (WHOIS, репутація IP/доменів, гео, дані з CMDB/asset-інвентаря, приналежність користувачів, критичність активів). Модуль пріоритизації застосовує політики ризику та ML-моделі (scoring за MITRE ATT&CK, історикою інцидентів, чутливістю активу), формуючи чергу роботи SOC. Движок оркестрації і плейбуків (playbook engine)

виконує графі дій із розгалуженнями, таймаутами, умовами та «людиною в циклі» (approval/consent) і керує виконавцями-робітниками (automation workers) для паралельних завдань. Підсистеми кейс-менеджменту та розслідувань об'єднують події в кейси, ведуть хронологію, артефакти й доказову базу; звітність і аналітика відстежують MTTD/MTTR, dwell time, частку автоматизованих дій, точність рішень ML. На рівні експлуатації важливі RBAC/ABAC-контроль, аудит усіх дій, секрет-сховище (vault), керування версіями плейбуків (GitOps), середовища dev/test/prod, черги/шини повідомлень для стійкості (наприклад, Kafka/AMQP), та HA/масштабування горизонтально.

Принцип роботи SOAR (етапи життєвого циклу інциденту)

Принцип роботи SOAR реалізується як безперервний цикл обробки інцидентів від моменту отримання події до формування висновків. На першому етапі тригер і інжест забезпечують надходження алерта із SIEM або сенсорів, таких як EDR чи NIDS, що одразу запускає відповідний плейбук за типом чи джерелом події.

Далі відбувається нормалізація, дедуплікація та кореляція, коли система уніфікує всі поля, прибирає дублікати й поєднує отримані події з уже наявними кейсами для створення цілісної картини.

Третій етап — автоматичне збагачення та класифікація, коли SOAR підтягує додатковий контекст із зовнішніх та внутрішніх джерел (репутаційні сервіси, CMDB, історія користувачів), оцінює ризик за допомогою ML-моделей і визначає пріоритетність інциденту.

Наступний крок — орієнтовна перевірка (triage), яка включає швидкі тести IOC у sandbox, перевірки YARA чи дані телеметрії EDR; у разі необхідності інцидент передається аналітику з готовим чек-лістом для уточнення.

Після цього виконуються оперативні дії containment, що можуть бути повністю автоматизовані: блокування IP чи домену на NGFW, ізоляція зараженого хоста у EDR, примусове завершення сесій, відключення компрометованого акаунта в IAM або оновлення політик поштового шлюзу.

Шостий етап включає ерадикацію і відновлення, коли видаляються артефакти, застосовуються патчі чи нові сигнатури, перевіряється цілісність систем, а сервіси повертаються в нормальний режим роботи у взаємодії з IT-відділом через ITSM. Завершальний етап передбачає lessons learned: автоматичне формування звітів, оновлення бази знань, оптимізацію плейбуків, а також навчання ML-моделей на нових даних для підвищення точності в майбутньому. Весь цикл супроводжується контрольними точками погодження для критичних дій і можливістю відкату (rollback), що мінімізує ризик помилок та побічних ефектів, роблячи SOAR ефективним і надійним інструментом автоматизації кіберзахисту.

Налаштування та оптимізація SOAR

Ефективність SOAR-системи значною мірою залежить від правильного налаштування та оптимізації, адже вона інтегрує десятки джерел подій і виконавчих інструментів у єдину екосистему. Першим кроком є конфігурація інтеграцій — підключення SIEM, EDR/XDR, IDS/IPS, NGFW, DLP, поштових шлюзів, CASB, сервісів IAM та ITSM. Для цього використовуються API-конектори, вебхуки або потокові шини (Kafka, AMQP).

Наступний етап — налаштування плейбуків (playbooks), які визначають автоматизовані сценарії реагування на інциденти. Вони створюються у вигляді графів дій із розгалуженнями, таймерами, перевітками умов і контрольними точками «людина в циклі» для критичних кроків. Це дозволяє балансувати між швидкістю автоматизації та контролем безпеки.

Для підвищення точності обробки подій застосовуються політики пріоритизації та ML-моделі. SOAR може автоматично призначати вагу інциденту залежно від критичності активу, контексту (наприклад, бізнес-функції сервера), історії атак чи тактик MITRE ATT&CK. Це забезпечує ефективне сортування та уникнення перевантаження SOC малозначущими алертами.

Важливим елементом налаштування є інтеграція з Threat Intelligence — зовнішніми та внутрішніми базами даних, які збагачують інциденти інформацією про репутацію IP, доменів, хешів файлів чи IOC. Це дозволяє автоматично приймати рішення про блокування без участі аналітика.

Оптимізація SOAR передбачає:

- регулярний перегляд і вдосконалення плейбуків,
- контрольні тести з імітацією атак (purple teaming),
- моніторинг ключових метрик (MTTD, MTTR, dwell time, % автоматизованих дій),
- впровадження GitOps-підходів для версійності плейбуків,
- забезпечення відмовостійкості та масштабування системи через горизонтальне розгортання.

Завдяки правильному налаштуванню SOAR перетворюється з простої «машини автоматизації» на адаптивний центр реагування, здатний зменшувати навантаження на SOC-аналітиків, підвищувати точність рішень і стандартизувати процеси.

Що робить SOAR сучасним

Платформи підтримують AI/ML для enrichment і авто-класифікації, MITRE ATT&CK-мапінг тактик і технік, автоматичне створення гіпотез (attack story-building) та симуляції «purple-team» сценарії для тесту плейбуків. Інтеграція з XDR забезпечує наскрізну видимість і спільні дії між поштою, ендпойнтами, ідентичністю, мережею та хмарами; з SASE/Zero Trust —

контекстні рішення з урахуванням стану користувача/пристрою. Для масштабних організацій критичні мульти-тенантність SOC, делегована адміністрація, каталог затверджених плейбуків і комплаєнс-шари (логування дій, цифрові підписи, SLA/OLA-метрики). Результат — скорочення MTDD/MTTR, зняття ручної рутини з аналітиків, повторюваність дій та контрольований, перевіряємий процес реагування у гібридних і мультихмарних середовищах.

Такі рішення дедалі частіше інтегруються з Big Data-аналітикою, що дозволяє корелювати інциденти з великими обсягами зовнішніх та внутрішніх даних. Сучасні SOAR-платформи також підтримують threat intelligence feeds для оперативного збагачення контексту та автоматичного оновлення правил реагування. Важливим трендом є застосування low-code/no-code інтерфейсів для створення плейбуків, що робить платформу доступнішою для спеціалістів різного рівня. Крім того, зростає роль хмарних SOAR-рішень, які забезпечують масштабованість і простіше впровадження у глобальних компаніях. Значна увага приділяється також захисту самих SOAR-платформ, адже компрометація інструмента оркестрації може призвести до масштабних наслідків. У перспективі очікується ширше впровадження генеративного AI, який допомагатиме створювати сценарії реагування та рекомендації в реальному часі, що ще більше підвищить швидкість і ефективність SOC.

Таблиця 4.2. Класичні SOAR vs сучасні AI-підсилені SOAR

Критерій	Класичні SOAR	Сучасні AI-підсилені SOAR
Автоматизація	Виконання статичних плейбуків, орієнтованих на задані сценарії	Динамічне формування плейбуків із використанням ML та AI для адаптивної відповіді
Класифікація інцидентів	Базується на сигнатурах та вручну заданих правилах	AI/ML-моделі автоматично класифікують події та прогнозують розвиток атак
Контекст і збагачення	Використання обмежених інтеграцій із threat intelligence	Автоматичне enrichment із багатьох джерел (TI, CMDB, гео, поведінкові моделі)
Ефективність SOC	Вимагає значної ручної роботи аналітиків	Зменшує ручну рутину, дозволяючи аналітикам фокусуватися на складних кейсах
Гнучкість	Жорстко структуровані сценарії, складні у зміні	Low-code/no-code інтерфейси, динамічне оновлення сценаріїв
Кореляція подій	Обмежена до попередньо визначених правил	Використання Big Data, AI та MITRE ATT&CK-мапінгу для глибшої кореляції

Критерій	Класичні SOAR	Сучасні AI-підсилені SOAR
Масштабованість	Переважно on-premises або локальні інтеграції	Хмарні та мультихмарні розгортання з гнучким масштабуванням
Зворотний зв'язок (feedback loop)	Мінімальний, залежить від ручного оновлення правил	Автоматичне навчання ML-моделей і тюнінг правил на основі результатів інцидентів
Рівень зрілості безпеки	Використовується як допоміжний інструмент реагування	Стає ядром адаптивних SOC і інтегрованих XDR-рішень

Таблиця наочно демонструє еволюцію систем SOAR від класичних рішень до сучасних AI-підсилених платформ. У класичних реалізаціях SOAR основний акцент робився на автоматизації рутинних дій через статичні плейбуки, які вимагали значного ручного втручання для оновлення правил і класифікації інцидентів. Це дозволяло скоротити час реагування, але не забезпечувало достатньої адаптивності до динамічного середовища кіберзагроз.

Сучасні AI-підсилені SOAR значно розширюють ці можливості завдяки інтеграції машинного навчання та штучного інтелекту. Вони здатні автоматично класифікувати події, прогнозувати розвиток атак і будувати гнучкі сценарії реагування відповідно до контексту. Завдяки AI, збагачення подій відбувається швидше й глибше — з урахуванням поведінкових моделей, геолокаційних даних, репутаційних фідів і зв'язку з MITRE ATT&CK.

Крім того, сучасні рішення забезпечують високу масштабованість у хмарних і мультихмарних середовищах, пропонують зручні low-code/no-code інтерфейси для створення сценаріїв і мають вбудований механізм feedback loop для автоматичного навчання моделей. Це робить AI-підсилені SOAR ключовим елементом адаптивних SOC та XDR-екосистем, де основна увага зміщується від ручної обробки інцидентів до проактивної аналітики й попередження атак.

Сучасні тенденції інтеграції SIEM та SOAR

Сьогодні SIEM та SOAR дедалі частіше поєднуються в єдині екосистеми безпеки. Якщо SIEM відповідає за глибокий моніторинг, аналіз і кореляцію подій, то SOAR бере на себе автоматизоване реагування та оркестрацію процесів. У результаті підприємства отримують не лише «центр моніторингу», а й «центр реагування» в одному рішенні. Завдяки цьому створюються сучасні Security Operation Centers (SOC), де більшість дій виконується автоматично або напівавтоматично, а аналітики зосереджуються на складних і нестандартних загрозах. Важливою сучасною тенденцією є використання Extended Detection and Response (XDR), які поєднують SIEM, SOAR та інші інструменти,

забезпечуючи єдине вікно для моніторингу, аналізу й автоматизації безпеки в мультимарних і гібридних середовищах.

Сучасні тенденції інтеграції SIEM та SOAR передбачають тісне впровадження AI/ML-моделей для автоматичної класифікації інцидентів і прогнозування розвитку атак. Розширюється підтримка MITRE ATT&CK-мапінгу, що дозволяє стандартизувати аналіз та реагування у глобальному масштабі. Зростає роль cloud-native SIEM/SOAR-рішень, які інтегруються з SASE і Zero Trust для забезпечення наскрізного моніторингу в розподілених середовищах. Популярності набуває застосування SOAR playbooks-as-code, коли сценарії реагування версіонуються й перевіряються за DevSecOps-підходами. Підприємства активно впроваджують автоматизовані механізми feedback loop, які дозволяють SIEM оновлювати правила та ML-моделі на основі досвіду SOAR. У перспективі інтегровані екосистеми SIEM+SOAR стануть ядром адаптивних SOC, де рішення прийматимуться максимально швидко й гнучко.

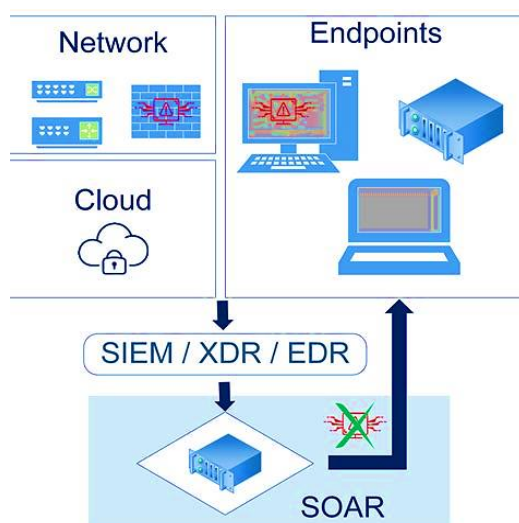


Рис. 4.24. Інтеграція SIEM, XDR та SOAR для централізованого моніторингу і реагування

Джерело: URL: <https://www.netacad.com/courses/cybersecurity/>

На рис. 4.24 зображено сучасну архітектуру взаємодії між SIEM, XDR/EDR та SOAR у системах кіберзахисту. Джерелами подій виступають мережа, хмарні сервіси та кінцеві точки, які передають дані у SIEM/XDR/EDR для збору, кореляції та виявлення загроз. Далі виявлені інциденти передаються до SOAR, що автоматизує реагування — блокує атаки, ізолює компрометовані пристрої чи облікові записи та знижує ризик поширення шкідливої активності. Такий підхід створює єдиний захисний контур, де SIEM забезпечує аналіз подій, XDR/EDR — глибоку видимість на рівні мережі й кінцевих точок, а SOAR — швидке реагування та усунення інцидентів.

Принцип роботи сучасної інтеграції SIEM та SOAR полягає у створенні єдиного замкненого циклу моніторингу й реагування, де обидва інструменти взаємодіють у реальному часі. SIEM виконує роль центрального вузла збору, нормалізації та кореляції подій безпеки з усіх джерел — серверів, мережевих пристроїв, хмарних сервісів і кінцевих точок. Після виявлення потенційної загрози SIEM формує алерт і передає його до SOAR, який автоматично запускає відповідний сценарій реагування (playbook). На цьому етапі система може виконати як повністю автоматизовані дії (блокування IP-адреси, ізоляція ендпойнта, відключення облікового запису), так і напівавтоматичні — з підтвердженням аналітика SOC. Завдяки впровадженню AI/ML-моделей SOAR може не лише автоматизувати рутинні завдання, але й оцінювати критичність інциденту, прогнозувати розвиток атаки та надавати рекомендації щодо оптимальних дій. Завершальний етап включає зворотний зв'язок: результати реагування надходять у SIEM для оновлення правил кореляції та навчання моделей, що забезпечує адаптивність та безперервне вдосконалення екосистеми кіберзахисту.

Значення для багаторівневого захисту

Роль SIEM і SOAR у сучасній архітектурі кіберзахисту полягає в тому, що вони забезпечують не лише видимість загроз, а й активну протидію їм. Завдяки централізованому збору й аналізу даних SIEM мінімізує ризик «сліпих зон» у корпоративній мережі, тоді як SOAR скорочує час реагування та оптимізує ресурси. Такий симбіоз дозволяє підприємствам оперативно виявляти, ізолювати та нейтралізувати атаки, що є критично важливим в умовах зростання кількості APT, Zero-Day та багаторівневих атак. У майбутньому ці системи дедалі більше використовуватимуть AI для створення передиктивної аналітики та повністю автономного реагування, що стане основою для побудови «самоозброєних» SOC наступного покоління.

На рис. 4.25 зображено, як дані з різних джерел (SIEM, API, email, threat intelligence feeds) надходять у хмарну платформу SOAR (наприклад, Sumo Logic). Усередині платформи видно модуляцію, де кореляційний двигун та ML-блок аналізують і збагачують події. Далі йде маршрутизація інцидентів через playbooks, які керують автоматичними діями (containment, ескалація), а також передають кейси до SOC для подальшого контролю. Ця схема добре відображає, як SOAR відіграє роль вузла оркестрації між розрізненими системами, AI-моделями та автоматичними сценаріями реагування.

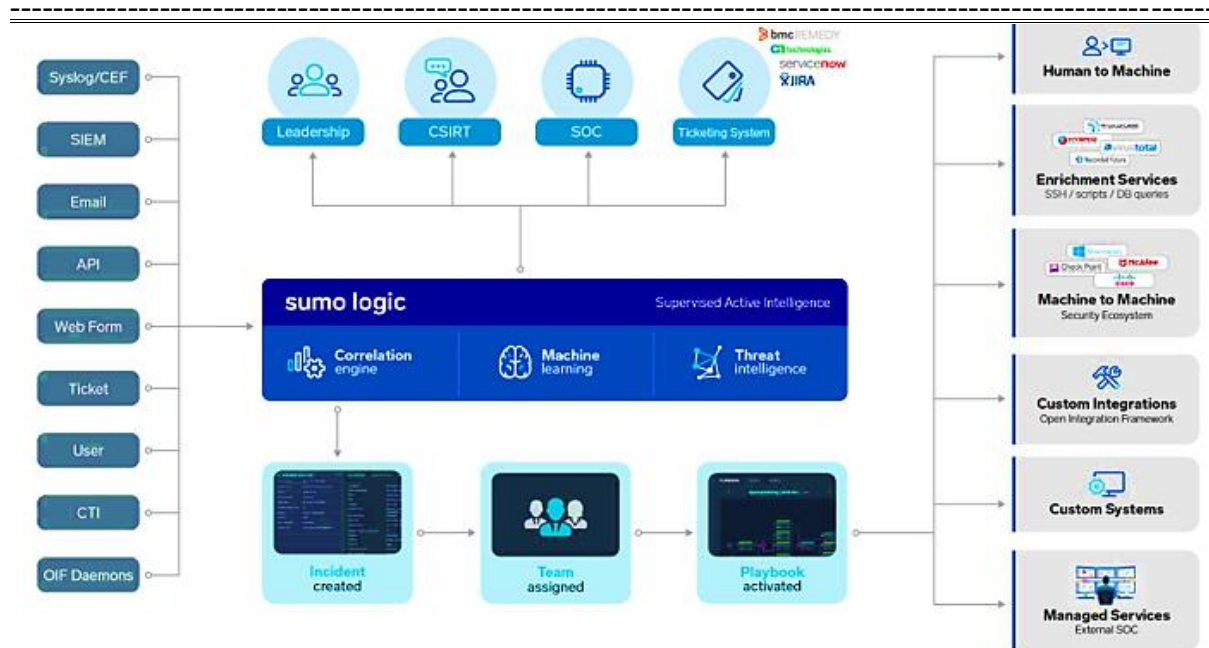


Рис. 4.25. Архітектура хмарного SOAR із кореляцією та AI/ML
Джерело: URL: <https://www.netacad.com/courses/cybersecurity/>

Значення для багаторівневого захисту полягає також у тому, що інтеграція SIEM та SOAR дозволяє поєднати профілактичний і реактивний підходи, забезпечуючи комплексне покриття всіх етапів життєвого циклу атаки. Використання цих платформ у зв'язці з IDS/IPS, NGFW, EDR та XDR створює єдиний захисний контур, де кожен рівень підсилює інший. Вони дають змогу формувати повноцінний ланцюг Kill Chain і швидко блокувати атаки ще на початкових фазах, зменшуючи ризик проникнення вглиб мережі. Важливим аспектом є їхня здатність інтегруватися з Zero Trust та SASE-моделями, що гарантує контроль доступу та перевірку довіри в реальному часі. Крім того, SIEM і SOAR відіграють ключову роль у забезпеченні комплаєнсу з міжнародними стандартами (ISO/IEC 27001, NIST, GDPR), автоматизуючи звітність і аудит. У результаті їхнє значення виходить далеко за межі класичного моніторингу й реагування — вони стають стратегічним інструментом для побудови стійкої кібербезпеки підприємства.

4.5. ЗАСОБИ ЗАХИСТУ КІНЦЕВИХ ТОЧОК: АНТИВІРУСНІ СИСТЕМИ, EDR/XDR

Засоби захисту кінцевих точок: антивірусні системи, EDR/XDR — це комплексні технології кібербезпеки, спрямовані на захист робочих станцій, серверів, мобільних пристроїв та інших кінцевих точок від атак, шкідливого програмного забезпечення та несанкціонованого доступу.

Антивірусні системи є базовим інструментом, що забезпечує виявлення та блокування відомих загроз за допомогою сигнатурного та евристичного аналізу. Вони формують перший рівень захисту, запобігаючи інфікуванню пристроїв класичними вірусами, троянами чи шпигунськими програмами.

EDR (Endpoint Detection and Response) розширює цей підхід, забезпечуючи постійний моніторинг, збір телеметрії з кінцевих точок і можливість реагувати на інциденти, включно з ізоляцією зараженого пристрою та проведенням цифрової криміналістики.

XDR (Extended Detection and Response) — це наступний рівень розвитку, який інтегрує дані не лише з кінцевих точок, але й із мереж, хмарних сервісів, поштових систем та ідентичностей користувачів. Він створює єдине середовище виявлення й реагування, застосовуючи кореляцію подій, AI/ML-моделі та автоматизацію для протидії складним багатовекторним атакам.

У сукупності ці засоби є критичними елементами багаторівневого захисту, поєднуючи превентивні, детективні та реактивні механізми для збереження цілісності та безпеки корпоративних інформаційних систем.

Антивірусні системи як базовий рівень захисту

Антивірусні рішення історично стали першою лінією захисту кінцевих точок (endpoints), таких як робочі станції, ноутбуки чи сервери. Їхній принцип роботи базується на виявленні шкідливого програмного забезпечення (вірусів, троянів, руткітів, шпигунських програм) за сигнатурами та евристичним аналізом. Сучасні антивірусні продукти значно еволюціонували — вони включають проактивні модулі для виявлення підозрілої поведінки, аналізу коду в режимі sandbox, сканування хмарних репутаційних баз та захисту від фішингових атак. Антивіруси залишаються критично важливими для базової гігієни безпеки, але в умовах зростання кількості Zero-Day експлойтів і складних атак їхніх можливостей недостатньо для комплексного захисту корпоративних середовищ.

Схема на рис. 4.26 показує, як клієнтський агент на кінцевій точці (ПК, сервер чи вузол) передає запити на сканування через HTTP або SMB до серверного компонента (CAVA/ICAP), який виконує глибоке сканування файлів. У робочому процесі клієнт передає файл чи його частину на сервер сканування, сервер аналізує його за допомогою антивірусного двигуна й повертає рішення (чистий, заражений, підозрілий). Якщо файл визнано шкідливим, система може відправити його в карантин або заборонити доступ, при цьому агент зберігає стан (мітки, час сканування, статус). Така архітектура підкреслює розподілений підхід — клієнти мінімізують навантаження, а ресурсоемна частина обробки виконується на сервері.

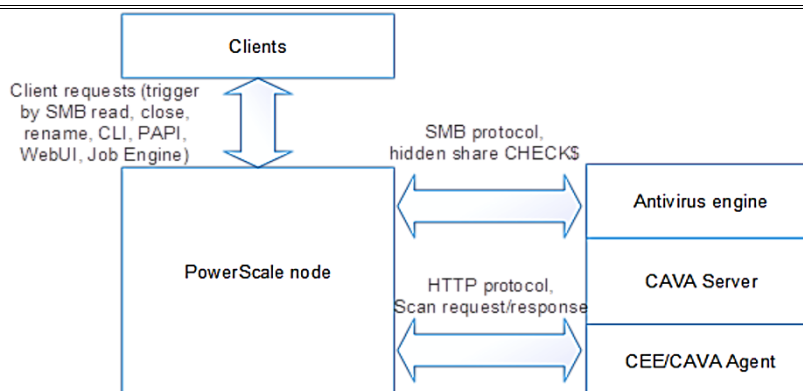


Рис. 4.26. Архітектура антивірусного рішення з клієнтським агентом і сервером сканування

Джерело: URL: <https://www.netacad.com/courses/cybersecurity/>

Антивірусні системи як базовий рівень захисту працюють за багаторівневим принципом: спочатку вони перевіряють файли й процеси на відповідність відомим сигнатурам, які зберігаються в базі даних загроз. Якщо співпадіння не знайдено, вмикається евристичний аналіз, що оцінює структуру коду та шукає підозрілі інструкції чи поведінкові шаблони. Додатково сучасні антивіруси використовують поведінкові модулі, які моніторять активність програм у реальному часі — наприклад, спроби несанкціонованого шифрування файлів або втручання у системні процеси. У випадку виявлення аномалій підозрілий об'єкт ізолюється у карантин для подальшого аналізу. Важливою функцією є інтеграція з хмарними репутаційними сервісами, що дозволяє миттєво отримувати дані про нові загрози з глобальних баз постачальника. Багато продуктів також вбудовують антифішингові фільтри, контроль веб-трафіку та поштових вкладень, а також захист від експлойтів у браузерях і офісних додатках. Незважаючи на значну еволюцію, антивірусні рішення залишаються лише початковим елементом багаторівневого захисту, адже складні цільові атаки (APT, Zero-Day) потребують додаткових технологій — EDR, XDR та інтеграції з SIEM/SOAR.

На схемі (рис. 4.27) видно кінцеву точку із агентом безпеки, який надсилає підозрілий файл до хмарного модуля sandbox для аналізу. У цьому модулі виконуються behavioral analysis і машинне навчання (ML), щоб спостерігати за поведінкою об'єкта в ізольованому середовищі. Паралельно відбувається збагачення даних через threat intelligence (репутаційні сервіси), які сприяють прийняттю рішення. Після інтеграції результатів sandbox, поведінкового та репутаційного модулів агент приймає рішення: дозволити, заблокувати або помістити в карантин. Це — сучасний приклад архітектури антивірусу з врахуванням усіх важливих компонентів для ефективного захисту кінцевих точок.

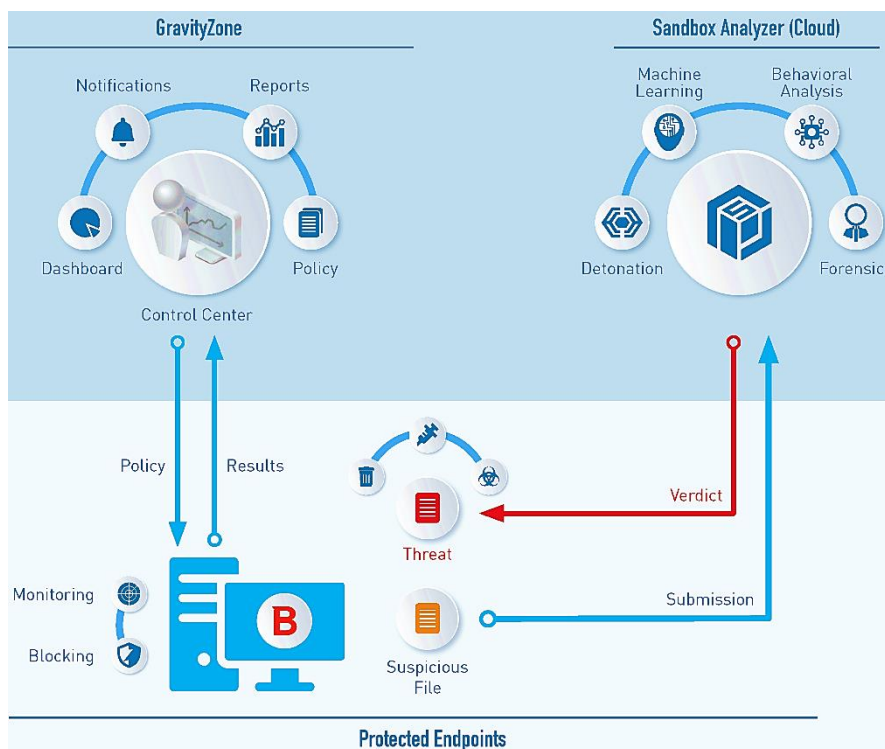


Рис. 4.27. Архітектура cloud-sandbox у антивірусному рішенні із поведінковим аналізом

Джерело: URL: <https://www.netacad.com/courses/cybersecurity/>

Налаштування та оптимізація антивірусних систем як базового рівня захисту

Антивірусні системи залишаються фундаментальним елементом кіберзахисту, забезпечуючи виявлення та блокування шкідливих програм на робочих станціях, серверах і мобільних пристроях. Правильне налаштування таких рішень дозволяє максимально підвищити їхню ефективність і знизити ризик проникнення загроз.

На етапі впровадження першочергово здійснюється повна інсталяція клієнтських агентів на всі кінцеві точки, включно з портативними пристроями та віддаленими робочими станціями. Агенти повинні бути інтегровані з єдиною консоллю управління, що дозволяє централізовано контролювати статус захисту, налаштовувати політики й отримувати звіти.

Важливим елементом є налаштування регулярного оновлення сигнатур і баз загроз. Оновлення повинні відбуватися автоматично з мінімальним інтервалом (щогодини або щодня), адже затримка навіть у кілька днів створює вікно вразливості для нових атак. Паралельно активуються механізми проактивного захисту (heuristic/behavioral analysis), які виявляють Zero-Day загрози за характером поведінки, навіть без наявності сигнатури.

Для оптимальної роботи антивірусної системи налаштовуються різні типи сканування:

- **On-access (реальний час)** — перевірка кожного файлу під час відкриття, завантаження чи виконання;
- **On-demand (ручне/за розкладом)** — повні сканування системи у визначений час, наприклад, уночі;
- **Boot-time scanning** — перевірка під час запуску ОС для виявлення прихованих загроз.

Особлива увага приділяється налаштуванню політик реагування: карантин, автоматичне видалення чи блокування підозрілих об'єктів. Для корпоративних середовищ критично важливо централізовано визначати, які дії мають виконуватися автоматично, а які — передаватися на підтвердження адміністратору.

Крім базових функцій, сучасні антивірусні системи підтримують інтеграцію з EDR/XDR-платформами, фаєрволами, SIEM та SOAR. Це дозволяє передавати алерти до SOC і запускати автоматизовані сценарії реагування (наприклад, ізоляцію зараженої машини).

Завершальний етап — оптимізація продуктивності: налаштування виключень для критично важливих бізнес-додатків, щоб уникнути зниження швидкодії, і регулярне тестування захисних політик через симуляції атак (наприклад, EICAR-тести або red teaming).

EDR (Endpoint Detection and Response)

EDR-системи стали відповіддю на нові виклики, оскільки вони не лише виявляють шкідливу активність, а й надають інструменти для детального аналізу інцидентів. Вони працюють за принципом безперервного моніторингу кінцевих точок, фіксують усі події (запуск процесів, зміни у файловій системі, мережеві підключення, реєстрові модифікації) та будують хронологію інциденту. EDR забезпечує можливість ретроспективного розслідування атак, ізоляції заражених пристроїв від мережі, збирання артефактів для криміналістики та інтеграції з SIEM/SOAR для автоматизації реагування. Сучасні EDR-рішення (Microsoft Defender for Endpoint, CrowdStrike Falcon, SentinelOne, VMware Carbon Black) активно використовують штучний інтелект і машинне навчання для виявлення аномалій і поведінкових патернів атак. Це дозволяє значно підвищити ефективність захисту від атак, що обходять класичний антивірус.

На рис. 4.28 показано, як кінцеві точки (endpoints) передають поведінкові дані (behavioral data) у центральну базу (Big Data Store / ML модулі). У цій центральній платформі відбувається аналіз, кореляція й збагачення інформації (information & context) із залученням алгоритмів машинного навчання. У разі

виявлення загрози система виконує автоматичне реагування (remediation) або надсилає сповіщення адміністраторам (manual). Схема ілюструє ключовий принцип EDR — безперервний моніторинг, поведінковий аналіз і автоматизоване реагування із залученням аналітиків при необхідності.

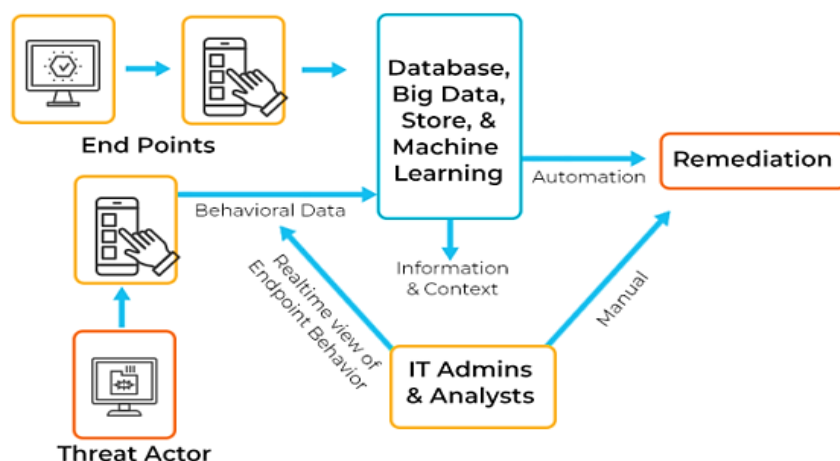


Рис. 4.28. Архітектура EDR з поведінковим аналізом і даними ML
Джерело: URL: <https://www.netacad.com/courses/cybersecurity/>

Принцип роботи EDR полягає в безперервному зборі телеметрії з усіх кінцевих точок, включно з активністю процесів, файловими операціями, підключеннями до мережі, діями користувачів і змінами в реєстрі чи конфігурації системи. Ці дані відправляються до центрального аналітичного модуля, де відбувається їхня нормалізація та кореляція. Далі застосовуються сигнатурні бази, поведінкові правила й алгоритми машинного навчання для виявлення підозрілих дій і відхилень від нормальної поведінки. У разі виявлення інциденту система може автоматично ініціювати реагування — ізолювати пристрій від мережі, завершити небезпечний процес, заблокувати файл або передати інформацію в SOAR для виконання плейбука. Важливо, що EDR підтримує можливість ретроспективного аналізу, що дозволяє дослідити ланцюг атаки навіть після її завершення, виявити «точку входу» і створити повну картину розвитку загрози.

Архітектура EDR зазвичай складається з декількох ключових рівнів. На кінцевих точках працюють агенти, які збирають усю необхідну телеметрію й передають її на сервер чи у хмарну платформу. Далі йде рівень збору та агрегації, що забезпечує централізоване сховище даних і попередню обробку інформації. На аналітичному рівні функціонують механізми кореляції, ML/AI-моделі, евристика та інструменти загрозової розвідки (Threat Intelligence), що збагачують дані й підвищують точність виявлення. Модуль реагування реалізує оперативні дії — від ізоляції вузлів і блокування процесів до генерації алертів у

SIEM чи запуску автоматизованих сценаріїв у SOAR. Завершальний рівень — це консолі для SOC-аналітиків, які забезпечують візуалізацію даних, пошук по всій базі телеметрії, побудову таймлайнів атак і створення звітів для керівництва та аудиту. Сучасні EDR усе частіше працюють у хмарі, що дозволяє масштабувати рішення й інтегрувати їх із XDR для наскрізного захисту в мультихмарних і гібридних інфраструктурах.

Налаштування та оптимізація EDR (Endpoint Detection and Response)

EDR-системи стали еволюційним продовженням класичних антивірусів, адже вони не лише виявляють шкідливе програмне забезпечення, а й забезпечують глибокий моніторинг поведінки кінцевих точок, аналітику телеметрії та активне реагування на інциденти. Вони є базовим елементом багаторівневого кіберзахисту корпоративних мереж, тому правильне налаштування і регулярна оптимізація EDR критично впливають на їхню ефективність.

Першим кроком у впровадженні EDR є розгортання агентів на всі кінцеві точки: робочі станції, сервери, мобільні пристрої та віддалені вузли. Важливо досягти повного покриття інфраструктури, адже будь-який неінтегрований пристрій стає потенційним вектором атаки. Для цього використовується централізована консоль управління, яка дозволяє контролювати статус агентів, оновлювати політики та відслідковувати інциденти.

Другим етапом виступає налаштування збору телеметрії. EDR отримує дані про системні журнали, процеси, мережеві з'єднання, файлові зміни та активність реєстру. На цьому рівні визначається баланс між глибиною моніторингу та навантаженням на систему, щоб уникнути перевантаження каналів зв'язку чи сховищ. Оптимальним рішенням є застосування профілів збору залежно від типу активу: для серверів, робочих станцій чи критичних вузлів мережі.

Наступним кроком є конфігурація політик виявлення і реагування. EDR дозволяє застосовувати як сигнатурні, так і поведінкові та аномальні методи детекції. Адміністратори визначають сценарії реагування: автоматичне блокування процесу, ізоляція вузла від мережі, завершення підозрілої сесії чи створення кейсу для подальшого аналізу. Для критично важливих дій у більшості рішень передбачено схему «людина в циклі», коли фінальне рішення ухвалює аналітик SOC.

Особливої уваги потребує інтеграція EDR з іншими компонентами інфраструктури безпеки. Сучасні рішення передбачають взаємодію з SIEM, SOAR, системами управління ідентичністю (IAM), NGFW і DLP. Це дозволяє автоматично передавати інформацію про інциденти в кореляційні та оркестраційні системи, які виконують автоматизовані playbooks. Такий підхід

значно скорочує час від виявлення до нейтралізації загрози (MTTR) та забезпечує єдину видимість усіх подій.

Важливим компонентом оптимізації є застосування штучного інтелекту та машинного навчання. Більшість сучасних EDR-платформ підтримують моделі UEBA (User and Entity Behavior Analytics), які будують профілі звичайної поведінки користувачів і виявляють аномальні відхилення. Це дозволяє виявляти Zero-Day атаки, внутрішні загрози та спроби компрометації облікових записів навіть без наявності сигнатур.

Ще одним аспектом є мінімізація помилкових спрацювань і впливу системи на бізнес-процеси. Для цього налаштовуються винятки для критично важливих застосунків, що не повинні блокуватися, та виконується регулярне тестування політик безпеки. Перевірка ефективності EDR здійснюється через red teaming, penetration testing або симуляції атак (BAS — Breach and Attack Simulation). Це дає змогу оцінити реальну готовність системи до інцидентів та своєчасно коригувати правила виявлення.

Завершальним етапом є постійне оновлення та масштабування EDR-платформи. Системи мають регулярно отримувати нові сигнатури, ML-моделі й сценарії реагування, а також забезпечувати гнучке горизонтальне масштабування у великих або мультихмарних середовищах. Усе це робить EDR не просто інструментом контролю кінцевих точок, а стратегічним компонентом сучасного SOC, який поєднує в собі можливості виявлення, реагування та глибокого аналізу кіберзагроз.

XDR (Extended Detection and Response)

XDR є еволюцією EDR, розширюючи його можливості на рівень усієї корпоративної екосистеми. Якщо EDR зосереджений на кінцевих точках, то XDR охоплює також мережу, поштові системи, хмарні середовища, ідентичності користувачів та IoT/OT-пристрої. Це забезпечує цілісний підхід до виявлення та реагування на атаки у різних векторах. Основна перевага XDR полягає в об'єднанні телеметрії з різних джерел у єдину платформу, де за допомогою кореляції подій, аналітики Big Data та AI/ML-моделей виявляються складні багатоступеневі атаки. XDR інтегрується з SIEM і SOAR, створюючи багаторівневий цикл захисту, де від виявлення загрози до реагування проходить мінімальний час. Провідні платформи XDR (Palo Alto Cortex XDR, Microsoft 365 Defender, Trend Micro Vision One, Fortinet FortiXDR) дозволяють будувати «єдине вікно» SOC, де аналітики бачать повний ланцюг атаки й можуть реагувати як автоматизовано, так і вручну.

Принцип роботи XDR базується на створенні єдиної екосистеми виявлення та реагування, де різні джерела телеметрії поєднуються у спільному середовищі аналізу. На першому етапі XDR отримує дані від кінцевих точок

(EDR-агенти), мережевих сенсорів (NDR), поштових шлюзів, хмарних API та систем управління ідентичностями. Далі виконується кореляція подій у режимі реального часу з використанням Big Data-платформ, де AI/ML-моделі аналізують поведінку, виявляють аномалії та будують цілісні ланцюги атак (attack storyline). На відміну від класичних EDR, XDR дозволяє бачити не окремий інцидент, а повну атаку, що проходить через кілька векторів (наприклад, фішингова пошта → компрометація акаунта → бічний рух у мережі → завантаження шкідливого ПЗ). Після виявлення загрози XDR запускає інтегровані механізми реагування — ізоляцію пристроїв, блокування користувацьких сесій, видалення шкідливих файлів, зміни політик доступу — при цьому можливе як повністю автоматичне виконання дій, так і погоджене з аналітиком SOC. Завдяки безперервному циклу зворотного зв'язку правила й ML-моделі постійно оновлюються, що підвищує точність виявлення у майбутньому.

Архітектура XDR являє собою багаторівневу інтегровану систему, у якій усі компоненти працюють як єдине ціле для забезпечення наскрізного захисту корпоративного середовища. На першому рівні функціонує шар збору даних (Data Ingestion Layer), який охоплює агентів на кінцевих точках, мережеві сенсори, поштові шлюзи, API-хмар та IoT/OT-конектори. Цей рівень відповідає за безперервний потік телеметрії, що надходить із різних джерел інфраструктури. Далі дані обробляються на шарі нормалізації та кореляції, де уніфікуються формати логів, усуваються дублікати, застосовується мапінг MITRE ATT&CK та відбувається побудова графів атак для відображення повної картини інциденту.

Ключову роль виконує аналітичний шар (Analytics Layer), де задіяні AI/ML-моделі, механізми поведінкового аналізу та UEBA (User and Entity Behavior Analytics). Саме тут відбувається виявлення аномалій, складних багатоступеневих атак та формування сценаріїв можливого розвитку подій. Наступним елементом є оперативний шар реагування (Response Layer), який реалізує автоматизовані сценарії реагування у вигляді playbooks та інтегрованих дій через SOAR. Він забезпечує ізоляцію заражених вузлів, блокування облікових записів, скидання сесій чи токенів, застосування патчів і навіть зміну політик безпеки у реальному часі.

Зручність роботи аналітиків гарантує інтерфейс SOC (Single Pane of Glass) — єдиний дашборд, який візуалізує всі інциденти, надає повний контекст і хронологію атаки та дозволяє керувати як автоматизованими, так і ручними діями. Завершує архітектуру інтеграційний шар, що забезпечує взаємодію з SIEM для довгострокового збереження й глибокого аналізу журналів, а також з SOAR для оркестрації автоматизованих сценаріїв реагування. Таким чином,

XDR створює цілісний, багаторівневий та адаптивний підхід до виявлення й протидії кіберзагрозам у складних гібридних і мультихмарних середовищах.

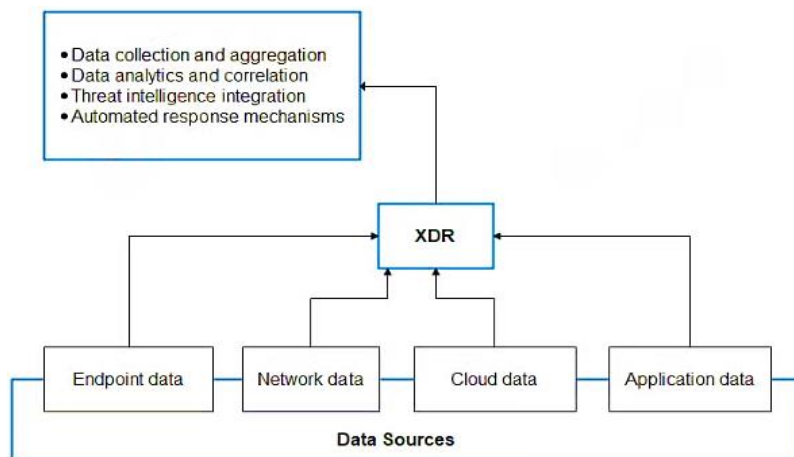


Рис. 4.29. Архітектура XDR: агрегація, кореляція, автоматичне реагування
Джерело: URL: <https://www.netacad.com/courses/cybersecurity/>

На рис. 4.29 зображено, як різні джерела даних — кінцеві точки (endpoints), мережевий трафік (network data) та хмарні сервіси — надходять у центральний модуль XDR через шар збору та агрегації даних. У самому модулі XDR відбувається нормалізація, кореляція подій і інтеграція з threat intelligence для збагачення контексту. Далі використовується аналітика (AI/ML-модулі) для виявлення аномалій, складних ланцюжків атак та пріоритизації інцидентів. Останній етап — механізми автоматичного реагування: ізоляція вузлів, блокування процесів, ескалація до SOAR або запуск playbooks, що реалізують усі необхідні дії. Така схема добре ілюструє, як XDR об'єднує телеметрію з різних доменів безпеки й автоматизує реагування у рамках єдиної платформи.

Налаштування та оптимізація XDR (Extended Detection and Response)

XDR (Extended Detection and Response) є еволюційним розвитком концепцій EDR та SIEM, який об'єднує моніторинг і реагування на інциденти з різних рівнів — кінцевих точок, мережевих сегментів, хмарних середовищ, електронної пошти та ідентифікаційних сервісів — у єдину інтегровану платформу. Основна мета XDR полягає у створенні централізованої картини загроз і автоматизації захисту завдяки кореляції подій у межах усієї інфраструктури підприємства.

Першим етапом впровадження XDR є інтеграція всіх джерел подій. На відміну від класичних EDR, XDR не обмежується лише кінцевими точками: до системи підключаються мережеві сенсори, поштові шлюзи, проксі-сервери, CASB, хмарні сервіси (AWS, Azure, Google Cloud), а також рішення IAM/SSO.

Це забезпечує єдиний канал збору телеметрії, який уніфікує дані про атаки та дозволяє бачити взаємозв'язки між різними векторами загроз.

Другим важливим кроком є налаштування аналітичного ядра. XDR використовує комбінацію правил кореляції, поведінкових моделей і ML/AI-алгоритмів для виявлення комплексних сценаріїв атак. Наприклад, одночасні підозрілі дії на пошті (фішинговий лист), мережевому рівні (підозрілі з'єднання) та кінцевій точці (старт аномального процесу) можуть бути об'єднані у єдиний інцидент. Для оптимізації роботи необхідно визначати базові профілі поведінки користувачів і систем, застосовувати UEBA (User and Entity Behavior Analytics) і формувати власні політики реагування.

Третій аспект налаштування XDR стосується автоматизації реагування. Система дозволяє створювати інтегровані сценарії (playbooks), які охоплюють декілька рівнів: блокування IP на брандмауері, ізоляція вузла у EDR, блокування поштового акаунта у системі IAM чи примусове завершення сесії у хмарному середовищі. Такий підхід значно скорочує MTTD (Mean Time to Detect) та MTTR (Mean Time to Respond), що критично важливо для підприємств з великою розподіленою інфраструктурою.

Наступним кроком є інтеграція XDR із SOC. Завдяки централізованим дашбордам аналітики отримують єдиний огляд інцидентів, які вже класифіковані, пріоритизовані та збагачені контекстом. Це зменшує кількість помилкових спрацювань і дає змогу фокусуватися на критичних загрозах. Оптимізація досягається шляхом тісної взаємодії XDR із SIEM: якщо SIEM відповідає за збереження та довгострокову аналітику даних, то XDR — за швидке виявлення і реагування у реальному часі.

Окремо варто виділити налаштування ML-моделей у XDR. Більшість рішень (Microsoft Defender XDR, Palo Alto Cortex XDR, Trend Micro Vision One) дозволяють навчати моделі на специфічних даних організації. Це включає адаптацію алгоритмів до бізнес-процесів, класифікацію активів за критичністю та оптимізацію політик ризику. В результаті XDR не тільки знижує кількість інцидентів, а й надає прогностичну аналітику для попередження атак.

Завершальним етапом оптимізації є постійний моніторинг і вдосконалення сценаріїв реагування. Адміністратори SOC мають регулярно оновлювати плейбуки, перевіряти їх ефективність під час Red Team/Blue Team-симуляцій, проводити аналіз lessons learned після кожного серйозного інциденту. Це дозволяє налаштовувати XDR як динамічну систему, здатну адаптуватися до нових векторів атак і мінімізувати час реагування навіть у мультихмарних чи гібридних середовищах.

Таким чином, XDR можна розглядати як «нервову систему» сучасної IT-безпеки: вона збирає сигнали з усього цифрового організму підприємства, аналізує їх у єдиному центрі та запускає автоматизовану реакцію. Грамотно

налаштована й оптимізована XDR-платформа значно підвищує стійкість підприємства до сучасних кіберзагроз і створює основу для побудови повноцінної Zero Trust-архітектури.

Сучасні тенденції розвитку

Сьогодні EDR і XDR дедалі більше інтегруються з Zero Trust-архітектурою та концепцією SASE, що забезпечує наскрізний захист у мультимарних і гібридних середовищах. Активно застосовується поведінкова аналітика користувачів і пристроїв (UEBA), що дозволяє виявляти інсайдерські загрози й аномалії навіть без наявності шкідливого коду. Впровадження штучного інтелекту забезпечує передиктивний аналіз і здатність зупиняти атаки ще на етапі їхнього формування. Особлива увага приділяється інтеграції з DevSecOps-процесами, що дозволяє автоматично перевіряти безпеку кінцевих точок під час розробки й розгортання нових систем. У перспективі EDR/XDR стануть невід'ємною частиною адаптивних SOC нового покоління, де поєднуюватиметься автономне виявлення, автоматизоване реагування та постійне навчання AI-моделей.

Водночас ці системи вже виходять за межі класичного моніторингу, пропонуючи функції глибокої телеметрії та аналітики ризиків у режимі реального часу. Вони здатні не лише ізолювати заражені вузли, а й автоматично ініціювати оновлення політик доступу, що робить захист більш динамічним і контекстно залежним. Сучасні рішення підтримують інтеграцію з threat intelligence-платформами, завдяки чому отримують актуальні індикатори компрометації та корелюють їх із внутрішніми подіями. Це дозволяє значно скоротити dwell time атак і підвищити ефективність роботи SOC-команд. У майбутньому очікується тісніше злиття EDR/XDR з технологіями автоматизованого тестування безпеки (BAS) та віртуальними аналітиками на основі ШІ. Такий підхід сприятиме формуванню єдиної самонавчальної екосистеми захисту, здатної адаптуватися до постійно мінливих кіберзагроз.

На рис. 4.30 показано концепцію проникнення XDR у архітектуру Zero Trust: вгорі вказані ключові домени — ідентичності, пристрої, мережа, робочі навантаження (workloads) — які постійно оцінюються та контролюються. Під ними зображені відповідні заходи контролю та модулі XDR (наприклад, поведінкова аналітика, моніторинг пристроїв, керування доступом), що підтримують стратегію нульової довіри. У центрі видно, як підозрілі дії чи аномалії в одному з доменів можуть запускати кореляцію та збір контексту через XDR. Такий підхід ілюструє, що довіра не встановлюється одноразово — всі компоненти системи задіяні в постійному аналізі й взаємному перевірці доступу та активності.

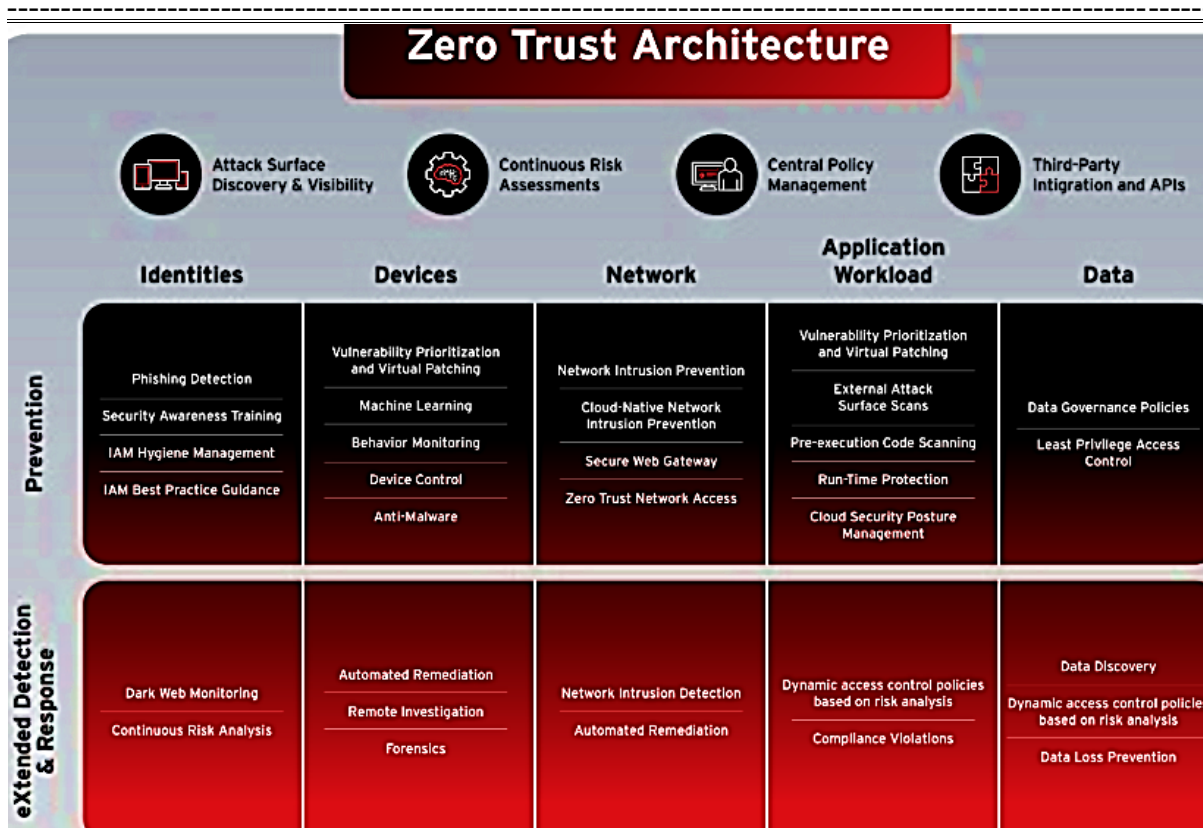


Рис. 4.30. Як Zero Trust та XDR працюють разом
Джерело: URL: <https://www.netacad.com/courses/cybersecurity/>

Сучасні тенденції розвитку EDR та XDR також включають активне використання хмарних репутаційних сервісів, які у реальному часі оновлюють дані про загрози та допомагають швидко блокувати нові вектори атак. Зростає популярність інтеграції з Mobile Threat Defense (MTD), що розширює можливості EDR/XDR на мобільні пристрої та IoT-середовище. Все більшого значення набуває автоматизована реакція на основі політик ризику, коли система самостійно обирає сценарій реагування залежно від критичності активу та впливу на бізнес-процеси. Водночас розвивається функціонал візуалізації ланцюгів атак, що дозволяє аналітикам SOC швидко зрозуміти весь перебіг інциденту у вигляді інтерактивних графів. Інтеграція з Threat Intelligence Platforms (TIP) забезпечує багатший контекст для аналізу, включаючи інформацію про кампанії APT та індикатори компрометації (IoC). У найближчій перспективі EDR/XDR перейдуть від реактивного захисту до повністю превентивних моделей, що забезпечуватимуть випереджальне блокування атак завдяки синергії AI, телеметрії та глобальних баз загроз.

Серед різновидів інтеграцій EDR та XDR особливе місце займає поєднання з SIEM та SOAR, яке забезпечує наскрізну обробку інцидентів від

збору логів до автоматизованого реагування. Таке з'єднання дозволяє створювати комплексні сценарії (playbooks), де дані з EDR/XDR стають тригерами для дій у брандмауерах, поштових шлюзах чи системах керування ідентичністю. Важливим напрямом є інтеграція з UEBA-платформами, що дозволяє відстежувати поведінку користувачів і пристроїв та виявляти приховані інсайдерські загрози. Не менш значущою є взаємодія з DevSecOps-інструментами, де EDR/XDR перевіряють безпеку застосунків і контейнерів уже на етапі CI/CD, знижуючи ризики впровадження вразливого коду. Додатково інтеграція з CASB-рішеннями та контролем доступу до хмарних ресурсів забезпечує уніфіковане управління безпекою у мультихмарному середовищі. Нарешті, взаємодія з Threat Hunting-платформами та системами автоматизованого тестування безпеки (BAS) дає змогу переходити від реактивного до проактивного підходу, де загрози не лише виявляються, а й передбачаються на основі симуляцій і аналітики великих масивів даних.

Різновиди інтеграцій EDR/XDR

Сучасні системи Endpoint Detection and Response (EDR) та Extended Detection and Response (XDR) вже не функціонують як ізольовані рішення. Вони інтегруються з широким спектром технологій кібербезпеки, утворюючи цілісну екосистему моніторингу, виявлення та реагування. Такий підхід дозволяє підвищити ефективність SOC, скоротити час реакції на інциденти й забезпечити глибший контекст для аналізу.

По-перше, ключовою є інтеграція з SIEM та SOAR. Дані з EDR/XDR слугують тригерами для кореляції в SIEM, а SOAR-рушії автоматизують реагування через плейбуки: блокування IP, ізоляція вузлів чи відключення акаунтів. Це створює повний цикл управління інцидентами — від збору логів до автоматизованої нейтралізації загроз.

По-друге, значного поширення набуває інтеграція з UEBA-платформами (User and Entity Behavior Analytics). Це дає змогу відстежувати поведінку користувачів і пристроїв, виявляти інсайдерські ризики та складні атаки без явного шкідливого коду.

По-третє, розвивається інтеграція з DevSecOps-інструментами. EDR/XDR контролюють безпеку контейнерів, мікросервісів та CI/CD-пайплайнів, запобігаючи впровадженню вразливого коду ще на етапі розробки. Це дозволяє організаціям впроваджувати безпеку «за замовчуванням» у життєвий цикл програмного забезпечення.

По-четверте, все частіше реалізується інтеграція з CASB-рішеннями (Cloud Access Security Broker). Це забезпечує контроль доступу до хмарних ресурсів, узгоджує політики доступу та дозволяє уніфікувати управління ризиками у мультихмарному середовищі.

По-п'яте, важливим напрямом є взаємодія з Threat Intelligence Platforms (TIP). Завдяки цьому EDR/XDR отримують актуальні індикатори компрометації (IoC), інформацію про АРТ-кампанії та нові вектори атак, що дозволяє підвищувати точність виявлення.

Нарешті, інтеграція з Threat Hunting-платформами та рішеннями BAS (Breach and Attack Simulation) переводить захист із реактивного у проактивний режим. SOC-аналітики можуть не лише реагувати на атаки, а й передбачати їх, тестуючи захист у режимі симуляцій.

Таким чином, різновиди інтеграцій EDR/XDR формують багаторівневу систему безпеки, де поєднуються автоматизація, поведінковий аналіз, аналітика загроз і управління ризиками, створюючи основу для адаптивного SOC нового покоління.

Різновиди інтеграцій EDR/XDR

Сучасні системи Endpoint Detection and Response (EDR) та Extended Detection and Response (XDR) функціонують не ізольовано, а в тісній взаємодії з іншими елементами екосистеми кіберзахисту. Саме завдяки інтеграціям вони забезпечують наскрізний моніторинг подій, автоматизацію реагування та формування повної картини загроз. Такий підхід дозволяє підприємствам підвищувати ефективність роботи SOC, скорочувати час виявлення й реагування на інциденти та переходити від реактивних до проактивних моделей захисту.

Одним із ключових напрямів є інтеграція EDR/XDR з SIEM та SOAR. SIEM отримує від EDR/XDR детальну телеметрію з кінцевих точок, мережевих сенсорів і хмарних сервісів, нормалізує її та корелює з іншими журналами подій. У свою чергу SOAR оркеструє процес реагування через попередньо визначені сценарії (playbooks), що включають блокування IP-адрес, ізоляцію скомпрометованих вузлів чи відключення облікових записів. У результаті формується повний цикл обробки інцидентів — від виявлення до автоматизованої нейтралізації.

Не менш важливим є поєднання з UEBA-платформами (User and Entity Behavior Analytics), які аналізують поведінку користувачів і пристроїв. EDR/XDR постачають технічні дані про логіни, процеси та підозрілі дії, а UEBA співставляє їх із поведінковими профілями, виявляючи приховані інсайдерські ризики чи компрометацію облікових записів. Така взаємодія підвищує точність виявлення атак, що не завжди містять шкідливий код.

В умовах розширеного використання хмар дедалі більшого значення набуває інтеграція EDR/XDR з CASB-рішеннями (Cloud Access Security Broker). Вона забезпечує контроль доступу до SaaS і хмарних платформ, дозволяючи оперативно блокувати скомпрометовані вузли або акаунти, які виявив

EDR/XDR. Це створює єдину політику управління доступом у мультихмарному середовищі.

Ще одним важливим напрямом є взаємодія з Threat Intelligence Platforms (TIP). Такі платформи постачають EDR/XDR актуальні індикатори компрометації (IoC), інформацію про АРТ-кампанії та нові техніки атак. Завдяки цьому система отримує додатковий контекст і здатна швидше виявляти та блокувати нові загрози.

Окремо слід виділити інтеграцію з BAS-рішеннями (Breach and Attack Simulation), які імітують реальні атаки з метою перевірки ефективності захисту. EDR/XDR у цьому випадку виступають «датчиками» й «виконавцями»: вони мають виявити атаки, згенеровані BAS, та активувати відповідні механізми реагування. Це дозволяє оцінити якість налаштувань і своєчасно оптимізувати захисні стратегії.

Таким чином, інтеграції EDR/XDR з іншими системами кіберзахисту формують багаторівневу й адаптивну інфраструктуру безпеки. Вони поєднують телеметрію, поведінковий аналіз, автоматизацію реагування та глобальну аналітику загроз у єдиному центрі управління, забезпечуючи підприємствам проактивний захист від сучасних кіберзагроз.

СПИСОК ДЖЕРЕЛ РОЗДІЛУ 4

1. Костюк Ю.В., Складанний П.М., Бебешко Б.Т., Хорольська К.В., Рзаєва С.Л., Ворохоб М.В. (2025) Безпека інформаційно-комунікаційних систем. Київський університет імені Бориса Грінченка, Україна, Київ. - 1017 с.
2. Костюк Ю.В., Складанний П.М., Гулак Г.М., Бебешко Б.Т., Хорольська К.В., Рзаєва С.Л. (2025) Системи захисту інформації. Київський університет імені Бориса Грінченка, Україна, Київ. - 886с.
3. Kostiuk, Yu. V., Skladannyi, P. M., Bebeshko, B. T., Khorolska, K. V., Rzaieva, S. L., & Vorokhob, M. V. (2025). Information and communication systems security. [Textbook] Kyiv: Borys Grinchenko Kyiv Metropolitan University.
4. Kostiuk, Yu. V., Skladannyi, P. M., Hulak, H. M., Bebeshko, B. T., Khorolska, K. V., & Rzaieva, S. L. (2025). Information security systems. [Textbook] Kyiv: Borys Grinchenko Kyiv Metropolitan University.
5. Гулак Г. М., Жильцов О. Б., Киричок Р. В., Коршун Н. В., Складанний П. М. Інформаційна та кібернетична безпека підприємства : підруч. / Г. М. Гулак, О. Б. Жильцов, Р. В. Киричок, Н. В. Коршун, П. М. Складанний – Львів : Видавець Марченко Т. В., 2023. – 370 с.

-
6. Hulak, Hennadii and Kriuchkova, Larysa and Skladannyi, Pavlo and Opirskyy, Ivan (2021) *Formation of Requirements for the Electronic Record-Book in Guaranteed Information Systems of Distance Learning* Cybersecurity Providing in Information and Telecommunication Systems 2021, 2963. pp. 137-142. ISSN 1613-0073
 7. Iosifova, Olena and Iosifov, Ievgen and Sokolov, V. Y. and Romanovskiy, O. and Sukaylo, I. (2021) *Analysis of Automatic Speech Recognition Methods* Cybersecurity Providing in Information and Telecommunication Systems, 2923. pp. 252-257. ISSN 1613-0073
 8. Zhebka, V. and Zhebka, S and Bazhan, T and Skladannyi, Pavlo and Sokolov, Volodymyr (2024) *Methodology for Choosing a Consensus Algorithm for Blockchain Technology* Digital Economy Concepts and Technologies Workshop 2024, 3665. pp. 106-113. ISSN 1613-0073
 9. Dudykevych, V. and Mykytyn, G. and Stosyk, T. and Skladannyi, Pavlo (2024) *Platform for the Security of Cyber-Physical Systems and the IoT in the Intellectualization of Society* Cybersecurity Providing in Information and Telecommunication Systems 2024, 3654. pp. 449-457. ISSN 1613-0073
 10. Shevchenko, Svitlana & Zhdanova, Yuliia & Shevchenko, Halyna & Nehodenko, Olena & Spasiteleva, Svitlana. (2023). Information Security Risk Management using Cognitive Modeling.
 11. V. Sokolov, P. Skladannyi and A. Platonenko, "Jump-Stay Jamming Attack on Wi-Fi Systems," 2023 *IEEE 18th International Conference on Computer Science and Information Technologies (CSIT)*, Lviv, Ukraine, 2023, pp. 1-5, doi: 10.1109/CSIT61576.2023.10324031.
 12. Buriachok, V., Korshun, N., Zhylytsov, O., Sokolov, V., Skladannyi, P. (2023). Implementation of Active Cybersecurity Education in Ukrainian Higher School. In: Faure, E., Danchenko, O., Bondarenko, M., Tryus, Y., Bazilo, C., Zaspá, G. (eds) *Information Technology for Education, Science, and Technics. ITEST 2022. Lecture Notes on Data Engineering and Communications Technologies*, vol 178. Springer, Cham. https://doi.org/10.1007/978-3-031-35467-0_32
 13. Marusenko, R., Sokolov, V., Buriachok, V. (2021). Experimental Evaluation of Phishing Attack on High School Students. In: Hu, Z., Petoukhov, S., Dychka, I., He, M. (eds) *Advances in Computer Science for Engineering and Education III. ICCSEEA 2020. Advances in Intelligent Systems and Computing*, vol 1247. Springer, Cham. https://doi.org/10.1007/978-3-030-55506-1_59
 14. V. Astapenya, V. Sokolov and D. Ageyev, "Experimental Evaluation of an Accelerating Lens on Spatial Field Structure and Frequency Spectrum," 2020 *IEEE Ukrainian Microwave Week (UkrMW)*, Kharkiv, Ukraine, 2020, pp. 203-206, doi: 10.1109/UkrMW49653.2020.9252755.

15. Brzhevska, Zoreslava and Kyrychok, Roman and Anosov, Andrii and Skladannyi, Pavlo and Vorokhob, Maksym (2021) *Analysis of the Process of Information Transfer from the Source-to-User in Terms of Information Impact* Cybersecurity Providing in Information and Telecommunication Systems II 2021, 3188 (2). pp. 257-264. ISSN 1613-0073
16. HU, ZHENGBING & Buriachok, Volodymyr & Sokolov, Volodymyr. (2019). Implementation of Social Engineering Attack at Institution of Higher Education. SSRN Electronic Journal. 10.2139/ssrn.3679106.
17. Mykhaylova, O and Fedynyshyn, T and Sokolov, Volodymyr and Kyrychok, Roman (2024) *Person-of-Interest Detection on Mobile Forensics Data—AI-Driven Roadmap* Cybersecurity Providing in Information and Telecommunication Systems 2024, 3654. pp. 239-251. ISSN 1613-0073
18. Marusenko, R., Sokolov, V., Skladannyi, P. (2023). Social Engineering Penetration Testing in Higher Education Institutions. In: Hu, Z., Dychka, I., He, M. (eds) *Advances in Computer Science for Engineering and Education VI. ICCSEE 2023. Lecture Notes on Data Engineering and Communications Technologies*, vol 181. Springer, Cham. https://doi.org/10.1007/978-3-031-36118-0_96
19. Mykhaylova, O and Fedynyshyn, T and Datsiuk, A and Fihol, B and Hulak, Hennadii (2023) *Mobile Application as a Critical Infrastructure Cyberattack Surface* Cybersecurity Providing in Information and Telecommunication Systems II 2023, 3050. pp. 29-43. ISSN 1613-0073
20. Kyrychok, R., Laptiev, O., Lisnevskiy, R., Kozlovskiy, V., & Klobukov, V. (2022). Development of a method for checking vulnerabilities of a corporate network using Bernstein transformations. *Eastern-European Journal of Enterprise Technologies*, 1(9(115)), 93–101. <https://doi.org/10.15587/1729-4061.2022.253530>
21. V. Sokolov, P. Skladannyi and V. Astapenya, "Bluetooth Low-Energy Beacon Resistance to Jamming Attack," *2023 IEEE 13th International Conference on Electronics and Information Technologies (ELIT)*, Lviv, Ukraine, 2023, pp. 270-274, doi: 10.1109/ELIT61488.2023.10310815.
22. Astapenya, Volodymyr & Zhdanova, Yuliia & Shevchenko, Svitlana & Spasiteleva, Svitlana & Kryvoruchko, Olena. (2024). Conflict Model of Radio Engineering Systems under the Threat of Electronic Warfare.
23. Dudykevych, V and Yevseiev, S and Mykytyn, H and Ruda, K and Hulak, Hennadii (2024) *Detecting Deepfake Modifications of Biometric Images using Neural Networks* Cybersecurity Providing in Information and Telecommunication Systems 2024, 3654. pp. 391-397. ISSN 1613-0073

24. Adamantis, M., Sokolov, V., Skladannyi, P. (2025). Evaluation of State-of-the-Art Machine Learning Smart Contract Vulnerability Detection Method. In: Hu, Z., Yanovsky, F., Dychka, I., He, M. (eds) *Advances in Computer Science for Engineering and Education VII. ICCSEEA 2024* 2024. Lecture Notes on Data Engineering and Communications Technologies, vol 242. Springer, Cham. https://doi.org/10.1007/978-3-031-84228-3_5
25. Chernenko, Roman and Anosov, Andrii and Kyrychok, Roman and Brzhevskaya, Zoreslava and Spasiteleva, Svitlana (2022) *Encryption Method for Systems with Limited Computing Resources* Cybersecurity Providing in Information and Telecommunication Systems 2022, 3288 (1). pp. 142-148. ISSN 1613-0073
26. Iavich, M. and Kovalchuk, O. and Gnatyuk, S. and Khavikova, Y. and Sokolov, Volodymyr (2024) *Classical and post-quantum encryption for GDPR* Classic, Quantum, and Post-Quantum Cryptography 2024, 3829. pp. 70-78. ISSN 1613-0073
27. Romanovskiy, O. and Iosifov, Ievgen and Iosifova, Olena and Sokolov, Volodymyr and Skladannyi, Pavlo and Sukaylo, Igor (2022) *Prototyping Methodology of End-to-End Speech Analytics Software* Modern Machine Learning Technologies and Data Science Workshop, 3312 (1). pp. 76-86. ISSN 1613-0073
28. Romanovskiy, O., Iosifov, I., Iosifova, O., Sokolov, V., Kipchuk, F., Sukaylo, I. (2021). Automated Pipeline for Training Dataset Creation from Unlabeled Audios for Automatic Speech Recognition. In: Hu, Z., Petoukhov, S., Dychka, I., He, M. (eds) *Advances in Computer Science for Engineering and Education IV. ICCSEEA 2021*. Lecture Notes on Data Engineering and Communications Technologies, vol 83. Springer, Cham. https://doi.org/10.1007/978-3-030-80472-5_3
29. Bessalov, A., Sokolov, V., & Abramov, S. (2024). Efficient Commutative PQC Algorithms on Isogenies of Edwards Curves. *Cryptography*, 8(3), 38. <https://doi.org/10.3390/cryptography8030038>
30. Syrotynskiy, R. and Tyshyk, I. and Kochan, O. and Sokolov, Volodymyr and Skladannyi, Pavlo (2024) *Methodology of network infrastructure analysis as part of migration to zero-trust architecture* Cyber Security and Data Protection 2024 (3800). pp. 97-105. ISSN 1613-0073
31. Martseniuk, Y. and Partyka, A. and Harasymchuk, O. and Shevchenko, Svitlana (2024) *Universal centralized secret data management for automated public cloud provisioning* Cybersecurity Providing in Information and Telecommunication Systems II 2024, 3826. pp. 72-81. ISSN 1613-0073

-
32. Iosifov, Ievgen and Iosifova, Olena and Sokolov, Volodymyr and Skladannyi, Pavlo and Sukaylo, Igor (2021) *Natural Language Technology to Ensure the Safety of Speech Information* Cybersecurity Providing in Information and Telecommunication Systems II. Volume I, 3187 (1). pp. 216-226. ISSN 1613-0073
33. Bahatskyi, Oleksii and Bahatskyi, V. and Sokolov, Volodymyr (2023) *Smart Home Subsystem for Calculating the Quality of Public Utilities* Cybersecurity Providing in Information and Telecommunication Systems, 3421. pp. 168-173. ISSN 1613-0073
34. V. Sokolov, P. Skladannyi and N. Mazur, "Wi-Fi Repeater Influence on Wireless Access," *2023 IEEE 5th International Conference on Advanced Information and Communication Technologies (AICT)*, Lviv, Ukraine, 2023, pp. 33-36, doi: 10.1109/AICT61584.2023.10452421.
35. Dreis, Y. and Korchenko, O. and Brzhevskaya, Zoreslava and Kriuchkova, Larysa and Nesterova, Olena (2024) *Model to Formation Data Base of Internal Parameters for Assessing the Status of the State Secret Protection* Cybersecurity Providing in Information and Telecommunication Systems 2024, 3654. pp. 277-289. ISSN 1613-0073
36. Virovets, Denis and Obushnyi, Sergiy and Zhurakovskiy, B. and Skladannyi, Pavlo and Sokolov, Volodymyr (2024) *Integration of smart contracts and artificial intelligence using cryptographic oracles* Classic, Quantum, and Post-Quantum Cryptography 2024, 3829. pp. 39-46. ISSN 1613-0073
37. Shevchenko, Svitlana & Zhdanova, Yuliia & Shevchenko, Halyna & Nehodenko, Olena & Spasiteleva, Svitlana. (2023). Conflict Analysis in the "Subject-to-Subject" Security System.
38. Myrontsov, Mykyta and Noskov, Oleksii and Sokolov, Volodymyr (2021) *Method for Determining Areas of Stable Solutions of Inverse Problems of Mathematical Physics by the Example of Well Electrometry* Information Technology and Mathematical Modeling for Environmental Safety 2021, 3021. pp. 115-126. ISSN 1613-0073
39. Buriachok, V., Hadzhyiev, M., Sokolov, V., Skladannyi, P., & Kuzmenko, L. (2019). Implantation of indexing optimization technology for highly specialized terms based on Metaphone phonetical algorithm. *Eastern-European Journal of Enterprise Technologies*, 5(2 (101), 43–50. <https://doi.org/10.15587/1729-4061.2019.181943>
40. Dreis, Yurii and Ivanichenko, Yevhen and Nesterova, Olena and Zhdanova, Yuliia and Dmytriienko, Kate (2022) *Restricted Information Identification Model* Cybersecurity Providing in Information and Telecommunication Systems 2022, 3288 (1). pp. 89-95. ISSN 1613-0073

-
41. Про основні засади забезпечення кібербезпеки України: Закон України // Відомості Верховної Ради України. 2017. № 45. Ст. 403.
 42. Закон України "Про інформацію".
 43. Закон України "Про захист інформації в інформаційно-комунікаційних системах".
 44. ДСТУ ISO/IEC 27001:2015 (ISO/IEC 27001:2013, IDT) "Інформаційні технології. Методи захисту. Системи управління інформаційною безпекою. Вимоги".
 45. ДСТУ ISO/IEC 27002:2015 "Інформаційні технології. Методи захисту. Практичні правила управління інформаційною безпекою".
 46. ДСТУ ISO/IEC 27005:2022 "Інформаційні технології. Методи захисту. Управління ризиками інформаційної безпеки".
 47. ДСТУ ISO/IEC 27701:2022 "Розширення для управління конфіденційністю в системах управління інформаційною безпекою".
 48. ДСТУ ISO/IEC 15408:2020 "Критерії оцінювання безпеки інформаційних технологій".
 49. ДСТУ ISO/IEC 24762:2015 "Інформаційні технології. Методи захисту. Рекомендації з відновлення інформаційно-комунікаційних технологій після інцидентів". Martseniuk, Yevhenii and Partyka, Andrii and Harasymchuk, Oleh and Cherevyk, Vyacheslav and Dovzhenko, Nadiia (2025) *Research of the Centralized Configuration Repository Efficiency for Secure Cloud Service Infrastructure Management* Cybersecurity Providing in Information and Telecommunication Systems 2025 (3991). pp. 260-274. ISSN 1613-0073
 50. Gnatyuk, S. and Sydorenko, V. and Polozhentsev, A. and Sokolov, Volodymyr (2024) *Method for managing IT incidents in critical information infrastructure facilities* Cybersecurity Providing in Information and Telecommunication Systems II 2024, 3826. pp. 326-333. ISSN 1613-0073
 51. Shapoval, V. and Zubyk, L. and Zubyk, Y. and Kurchenko, O. and Kozachok, Valerii (2024) *Automation of Data Management Processes in Cloud Storage* Cybersecurity Providing in Information and Telecommunication Systems 2024, 3654. pp. 410-418. ISSN 1613-073
 52. Dovzhenko, N. and Haidur, H. and Brzhevskaya, Zoreslava and Ivanichenko, Yevhen and Nesterova, Olena (2023) *Method of Sensor Network Functioning under the Redistribution Condition of Requests between Nodes* Cybersecurity Providing in Information and Telecommunication Systems 2023, 3421. pp. 278-283. ISSN 1613-0073
 53. Kulibaba, Serhii and Popereshnyak, Svitlana and Shcheblanin, Yuri and Kurchenko, Oleg and Mazur, Nataliia (2022) *Advanced Communication Model with the Voice Control and the Increased Security*

Level Cybersecurity Providing in Information and Telecommunication Systems 2022, 3288 (1). pp. 64-72. ISSN 1613-0073

54. Zybin, Serhii and Khoroshko, Volodymyr and Khokhlachova, Yuliia and Kozachok, Valerii (2021) *Approach of the Attack Analysis to Reduce Omissions in the Risk Management Cybersecurity Providing in Information and Telecommunication Systems* 2021, 2963. pp. 318-328. ISSN 1613-0073

55. V. Sokolov, P. Skladannyi and V. Astapenya, "Wi-Fi Interference Resistance to Jamming Attack," *2023 IEEE 5th International Conference on Advanced Information and Communication Technologies (AICT)*, Lviv, Ukraine, 2023, pp. 1-4, doi: 10.1109/AICT61584.2023.10452687.

56. Shevchenko, Svitlana and Zhdanova, Yuliia and Kryvytska, O. and Shevchenko, H. and Spasiteleva, Svitlana (2024) *Fuzzy cognitive mapping as a scenario approach for information security risk analysis Cybersecurity Providing in Information and Telecommunication Systems II* 2024, 3826. pp. 356-362. ISSN 1613-0073

57. Pinchuk, Olga and Sokolyuk, Oleksandra and Burov, Oleksandr and Lavrov, Evgeniy and Shevchenko, Svitlana and Aksakovska, Valeriia (2020) *ICT for training and evaluation of the solar impact on aviation safety* ICT in Education, Research, and Industrial Applications. Proc. 16 th Int. Conf. ICTERI 2020. Volume II: Workshops. Kharkiv, Ukraine, October 6-10, 2020, CEUR-WS.org., pp. 786-792. ISSN 1613-0073

Контрольні питання

1. Чим принципово відрізняються моделі DAC, MAC, RBAC та ABAC, і в яких випадках кожна з них є доцільнішою для підприємства?
2. Як Zero Trust доповнює класичні моделі контролю доступу і які його обов'язкові компоненти (ідентичність, мікросегментація, оцінка контексту)?
3. Наведіть приклад політики ABAC, що враховує атрибути користувача, ресурсу та середовища (час/гео/пристрій), і поясніть її переваги над RBAC.
4. Порівняйте ризики та операційні витрати впровадження MAC і RBAC у фінансовій установі з вимогами суворого поділу доступу.
5. Які типові помилки призводять до уразливостей у DAC та як їх пом'якшити (least privilege, MFA, аудит, SIEM-кореляція)?
6. Поясніть різницю між статичним (packet filtering), динамічним (stateful) та NGFW-фаєрволом на прикладі блокування сканування портів і прикладних атак.

7. Яку роль відіграє DPI та ідентифікація додатків в NGFW, і чому це критично для контролю трафіку в мультихмарних середовищах?
8. Опишіть багаторівневу стратегію, у якій статичні, stateful та NGFW взаємодіють із IDS/IPS та SIEM для зниження ризиків.
9. Які виклики виникають при захисті зашифрованого трафіку (TLS/SSL inspection) і які компроміси між приватністю, продуктивністю та видимістю?
10. Як інтеграція фаєрволів із SASE/XDR підсилює політики Zero Trust у розподілених (віддалених) робочих середовищах?
11. Чим відрізняються IDS і IPS з точки зору розміщення (passive vs inline), дій (alert vs block) та впливу на мережеву продуктивність?
12. Порівняйте HIDS і NIDS: які типи атак кожен виявляє краще та як їх комбінувати для виявлення латерального руху в мережі?
13. Які підходи сприяють зниженню хибнопозитивних спрацьовувань в IDS (базова лінія, тюнінг правил, ML/аналітика, кореляція подій)?
14. Опишіть життєвий цикл розгортання IPS у режимі prevention: від «alert-only»/тіньового режиму до поетапного ввімкнення блокування.
15. Як Big Data/streaming-аналітика (Kafka, Spark) та розподілені сенсори покращують масштабованість та точність хмарних NIDS/IPS?
16. Поясніть відмінності між SIEM та SOAR і чому їх зв'язка є «мозком і руками» SOC.
17. Опишіть типовий playbook у SOAR для інциденту «підозріла ексфільтрація даних»: які кроки автоматизуються, де потрібне підтвердження аналітика?
18. Які метрики доречно використовувати для оцінювання зрілості процесів моніторингу та реагування (MTTD, MTTR, dwell time) і як їх знижує SIEM+SOAR?
19. Як інтеграція SIEM/SOAR з MITRE ATT&CK і Zero Trust підсилює пріоритизацію інцидентів і відповідність стандартам (ISO/IEC 27001, NIST)?
20. Чим відрізняються антивірус, EDR і XDR за джерелами телеметрії, можливостями розслідування/ізоляції та рівнем кореляції подій у мультивекторних атаках?

РОЗДІЛ 5

КРИПТОГРАФІЧНІ МЕТОДИ ЗАБЕЗПЕЧЕННЯ БЕЗПЕКИ: АЛГОРИТМИ ШИФРУВАННЯ, ПРОТОКОЛИ ЗАХИЩЕНОГО ОБМІНУ ТА ПЕРСПЕКТИВИ ПОСТКВАНТОВОЇ КРИПТОГРАФІЇ



- 5.1. Математичні основи криптографії та їх значення для захисту даних**
- 5.2. Симетричні алгоритми шифрування (AES, ChaCha20, Twofish) та їх застосування**
- 5.3. Асиметричні алгоритми (RSA, ECC, постквантові методи) у захисті комунікацій**
- 5.4. Хеш-функції, цифрові підписи і контроль цілісності інформації**
- 5.5. Протоколи безпечного обміну даними (SSL/TLS 1.3, QUIC, IPsec, WireGuard)**
- 5.6. Сучасні напрями розвитку криптографії: квантова та гомоморфна криптографія**

Криптографічний стек сучасних інформаційних систем складається з трьох взаємопов'язаних шарів: симетричне шифрування для захисту великих обсягів даних і приватних каналів, асиметричні (публічно-ключові) механізми для встановлення ключів і цифрових підписів, а також протоколи поверх цих примітивів (TLS, SSH, IPsec), які реалізують повноцінні сервіси конфіденційності, цілісності й автентифікації. Симетричні алгоритми (наприклад, AES і ChaCha20) залишаються основою для потокового й файлового шифрування завдяки ефективності й відносній стійкості, тоді як асиметричні

схеми (RSA, Diffie–Hellman, ECC) забезпечують безпечний обмін симетричними ключами та підписи, що дозволяє будувати довірчі інфраструктури й сертифікаційні ланцюжки.

Критичною властивістю архітектури є розділення ролей: симетричні шифри відповідають за конфіденційність каналу після того, як ключ встановлено, а асиметричні примітиви — за безпечну доставку й автентифікацію цього ключа та за перевірку автентичності суб'єктів. На практиці сучасні протоколи використовують AEAD-схеми (Authenticated Encryption with Associated Data) для одночасного забезпечення цілісності й конфіденційності та проєктують механізми «криптоагільності», щоб можна було змінювати алгоритми без радикальної переробки протоколу або інфраструктури.

Однак поява квантових обчислень породжує новий вимір ризику: існують відомі квантові алгоритми, які теоретично руйнують безпеку класичних публічно-ключових схем. Зокрема, алгоритм Шора дозволяє ефективно розв'язувати задачі факторизації й дискретного логарифма — основи RSA і ECC — на достатньо потужному квантовому комп'ютері, а алгоритм Гровера дає квадратичне прискорення для «грубої сили», що знижує ефективну стійкість симетричних ключів, тому для довготривалого збереження конфіденційності рекомендують підвищувати довжину симетричних ключів (наприклад, AES-256 дає значний запас щодо Grover-атак).

У відповідь на ці загрози спільноти криптографів і стандартизатори (NIST) провели багаторічний процес відбору постквантових алгоритмів; як результат, до набору рекомендованих для стандартизації були відібрані алгоритми на основі решіток та інші конструкції (CRYSTALS-KYBER для встановлення ключів; CRYSTALS-DILITHIUM, FALCON, SPHINCS+ — для підписів; а згодом і додаткові кандидати) — це дає практичну основу для впровадження PQC у реальні системи і вже породило перші робочі специфікації й стандарти. Водночас вибір і впровадження нових алгоритмів вимагає ретельного тестування на сумісність, продуктивність і вразливість до нових класів криптоаналізу.

Практична міграція до постквантової ери вимагає декількох взаємопов'язаних підходів. По-перше, криптоагільність — здатність системи швидко змінювати набори алгоритмів і параметрів — має стати архітектурною вимогою: це означає модульні реалізації криптографії, схеми оновлення ключів і ретельно відпрацьовані процедури керування життєвим циклом ключів і сертифікатів. По-друге, в коротко/середньостроковій практиці рекомендовано застосовувати гібридні конструкції (одночасне використання класичного й постквантового KEX у протоколах, наприклад у TLS) — таким чином зберігається захист, навіть якщо один з компонентів виявиться зламаним. IETF і практичні реалізатори вже описують та впроваджують підходи до гібридного KEX у TLS 1.3, щоб забезпечити плавний перехід.

Технічні рекомендації для практиків зводяться до кількох взаємозалежних кроків: впровадити AEAD-шифри й сучасні параметри (наприклад, рекомендовані розміри ключів для симетрії), забезпечити можливість додавання/перемикання постквантових алгоритмів у TLS-стек, тестувати продуктивність і сумісність PQC-реалізацій у реальному навантаженні, а також організувати управління ключами й сертифікатами так, щоб знизити ризик «collect-now-decrypt-later» (тобто злочинці не могли сьогодні зберегти трафік і згодом розшифрувати його квантовим комп'ютером). Одночасно необхідно розвивати процедури збереження й аудитування криптографічних матеріалів: плани міграції мають містити тестові середовища, roll-back-механізми і чіткі контрольні точки.

Нарешті, переходячи до постквантових рішень, важливо пам'ятати про практичні компроміси: деякі PQC-алгоритми потребують більших ключів або породжують більш об'ємні підписи, що впливає на пропускну здатність і latency; також потрібно контролювати взаємодію з існуючими PKI і відпрацьовувати політики підтримки backward compatibility. Через це план міграції повинен поєднувати технічні заходи (гібридні KEX, збільшення симетричних параметрів, AEAD), операційні практики (crypto-lifecycle, тестування, моніторинг) і політичні/юридичні аспекти (retention політик, вимоги до довготривалого захисту даних). У підсумку, поєднання криптоагільності, гібридних протоколів і чіткої дорожньої карти до постквантових стандартів дає найбільш реалістичний і ризик-орієнтований шлях до збереження довготривалої конфіденційності й автентичності в епоху квантових обчислень.

5.1. МАТЕМАТИЧНІ ОСНОВИ КРИПТОГРАФІЇ ТА ЇХ ЗНАЧЕННЯ ДЛЯ ЗАХИСТУ ДАНИХ

Математичні основи криптографії та їх значення для захисту даних — це сукупність математичних теорій, алгоритмічних властивостей і складніших припущень, які лежать в основі всіх криптографічних примітивів і протоколів та визначають їхню безпеку і практичну застосовність. На найнижчому рівні це теорія чисел і абстрактна алгебра (модулярна арифметика, групи, кільця, поля), що дає формальний апарат для класичних публічно-ключових схем (RSA базується на властивостях факторизації великих цілих, схеми на основі дискретного логарифма — на поведінці експоненціації в групах, а еліптичні криві використовують арифметику над полями, де «важливість» поля й форми кривої впливає на складність обчислення дискретного логарифма). Ці математичні конструкції формалізують операції, які легко виконуються у прямому напрямку, але (за певних припущень) алгоритмічно важкі в зворотному — і саме ця «одностороння» властивість є ядром криптографічної стійкості.

Другим важливим блоком є математичні підходи до генерації псевдовипадковості та аналізу ентропії: теорія випадкових процесів, статистика і теорія інформації (ентропія, відстані між розподілами) використовуються для оцінки сили ключів, стійкості потокових шифрів і стійкості шифрування до статистичних атак. На практиці криптографічні протоколи спираються на генератори випадкових чисел з високою ентропією і на теоретичні гарантії псевдовипадковості (PRG/PRF/PRP), які формують основу для безпечної генерації ключів, ініціалізаційних векторів і некорельованих сесійних параметрів.

Третій пласт — теорія складності і формальна безпека: поняття складності обернених задач (NP-hard, поліноміальна/експоненційна складність), твердження про «складність» класичних задач (факторизація, дискретний логарифм) і сучасні підходи до доведень безпеки через редукції (показати, що злам схеми впливає з вирішення якоїсь важкої задачі). Теорія доказів у криптографії (симуляторні моделі, моделі довіри, інтенсивність вбудованих припущень) дає формальні гарантії та межі застосовності примітивів у протоколах. Разом із цим важливим є розуміння моделей атак — від класичних алгоритмічних до побічних каналів — бо математична стійкість без практичного моделювання середовища виконання може виявитися недостатньою.

Окрему роль відіграють математичні структури, що лежать в основі постквантових підходів: теорія решіток (lattice-based cryptography), кодування (code-based), мультиваріантні рівняння й хеш-орієнтовані підписи. Кожен клас має свої математичні припущення про складність (наприклад, складність короткого вектору в решітці), і їхнє використання дозволяє будувати примітиви, стійкі до атак з боку квантових комп'ютерів. Розуміння цих математичних основ необхідне для оцінки компромісів (розміри ключів, швидкість, обсяг підписів) при плануванні міграцій до постквантових рішень.

З практичної точки зору значення математичних основ для захисту даних багатоаспектне: по-перше, вони визначають якісні гарантії — які саме припущення необхідні, щоб конфіденційність/цілісність/автентичність була забезпечена; по-друге, вони встановлюють параметричні вимоги (довжини ключів, розміри модулів, параметри еліптичних кривих, глибина вирізу в решітках) для досягнення бажаної криптографічної сили; по-третє, вони формують критерії для вибору примітивів у конкретних системах з урахуванням продуктивності й ресурсних обмежень.

На рис. 5.1 подано інфографіку у вигляді трьох поруч розташованих блоків (синій — симетричні алгоритми, помаранчевий — асиметричні/публічно-ключові, зелений — постквантова криптографія), кожен із яких стисло описує: типові алгоритми, ключові властивості і практичні сценарії застосування. Лівий

блок підкреслює, що симетричні примітиви (AES, ChaCha20, потокові шифри) відзначаються високою продуктивністю і застосовуються для захисту каналів і зберігання даних, але вимагають надійного керування ключами і уваги до nonce/IV. Середній блок пояснює роль публічно-ключових схем (RSA, DH/ECDH, ECC): вони спрощують безпечний обмін ключами та реалізацію підписів і PKI, проте мають вищі обчислювальні витрати і залежать від математичних припущень (факторизація, дискретний логарифм). Правий блок присвячено постквантовим сімействам (решіткові, code-based, hash-based), їхнім властивостям (іноді великі ключі/підписи) і ролі в міграції до стійких до квантових атак рішень (гібридні KEX у TLS, довготривале шифрування). Інфографіка служить стислим наочним резюме для слайду або вступного розділу: вона показує, як різні математичні підґрунтя і практичні вимоги визначають вибір примітивів і архітектур (симетрія для сесій, асиметрія для KEX/підписів, PQC для захисту у довгостроковій перспективі).

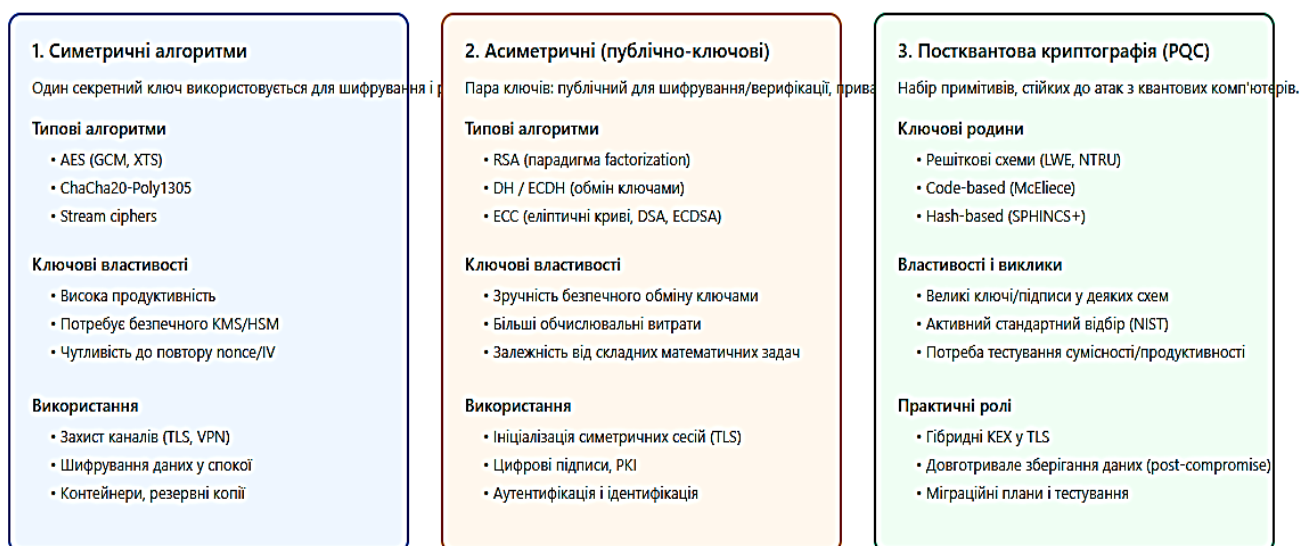


Рис. 5.1. Огляд криптографії — симетрична, асиметрична та постквантова категорії

Джерело: URL: <https://www.netacad.com/courses/cybersecurity/>

Нарешті, освітня й операційна складова: знання математичних основ дає змогу інженерам адекватно оцінювати ризики (наприклад, небезпеку «collect-now-decrypt-later»), впроваджувати принципи криптоагільності, коректно реалізовувати та параметризувати бібліотеки й протоколи і розуміти межі формальних доведень. Воно також указує на головний урок безпеки: навіть найміцніша математична гарантія втратить силу при неправильній реалізації, поганому управлінні ключами чи ігноруванні побічних каналів — отже,

математична теорія і інженерні практики мають працювати разом, щоб забезпечити реальний захист даних.

5.2. СИМЕТРИЧНІ АЛГОРИТМИ ШИФРУВАННЯ (AES, CHACHA20, TWOFISH) ТА ЇХ ЗАСТОСУВАННЯ

Симетричні алгоритми шифрування — це клас криптографічних примітивів, у яких для шифрування й розшифрування використовується один і той же секретний ключ; їхня роль у практичній безпеці — захист конфіденційності й, у поєднанні з механізмами цілісності/автентифікації, забезпечення цілісності та автентичності даних при зберіганні й передачі. Серед найбільш поширених і перевірених на практиці реалізацій виділяють три сімейства: AES, ChaCha20 (з Poly1305) та Twofish — кожне має свої математичні властивості, архітектурні обмеження й типові сценарії застосування.

У практичних впровадженнях симетричні шифри завжди використовуються в конкретних режимах роботи (блокові режими або як потокові шифри), що визначають їхню стійкість до певних класів атак та вплив на продуктивність. Сучасні рекомендації наполягають на застосуванні AEAD-режимів (наприклад, AES-GCM або ChaCha20-Poly1305), оскільки вони поєднують шифрування з автентифікацією і зменшують ризик помилкових поєднань шифрування й MAC.

Критичний операційний аспект — коректна робота з nonce/IV: повторне використання одного й того ж nonce з тим самим ключем може призвести до серйозних компрометацій, тому генерація й управління унікальними ініціалізаційними векторами має бути гарантована. На рівні продуктивності AES отримує значний вигравш завдяки апаратній підтримці (AES-NI), тоді як ChaCha20 показує кращі результати в програмних реалізаціях на платформах без такого прискорення, тому вибір часто залежить від цільової апаратної платформи та вимог до латентності. Невід'ємною частиною безпеки є правильний менеджмент ключів — генерація з високою ентропією, зберігання в HSM/KMS, регулярна ротація, розмежування доступу і чіткі політики ретеншну. Урахування цих інженерних практик разом з регулярними аудитами реалізацій (включно з тестуванням на побічні канали й використанням перевірених криптографічних бібліотек) робить симетричні примітиви ефективним і практично безпечним інструментом захисту даних.

AES (Advanced Encryption Standard)

AES (Advanced Encryption Standard) — блоковий шифр із розміром блоку 128 біт і ключами 128/192/256 біт, який став промисловим стандартом

для більшості протоколів і систем. У реальних застосунках AES зазвичай працює не «сам по собі», а в режимах шифрування: для захищеного обміну й поточних сценаріїв переважно використовують AEAD-режими (Authenticated Encryption with Associated Data) — наприклад, AES-GCM, що одночасно дає конфіденційність і цілісність, або для специфічних випадків — CTR+HMAC (старіше). AES дуже вигідно апаратно прискорюється на сучасних CPU завдяки інструкціям AES-NI, тому він часто є найшвидшим варіантом у серверах і в обладнанні, яке підтримує апаратне прискорення. У виборі режиму й параметрів важливо дотримуватися рекомендацій: наприклад, для AES-GCM рекомендується 96-бітовий попсе (уникаючи повторного використання попсе для одного ключа), а для дискових шифрувань застосовується спеціальний режим XTS-AES (призначений для сховищ), при цьому XTS не надає автентифікації даних і тому слід обирати його свідомо з урахуванням ризиків.

AES (Advanced Encryption Standard) — це сучасний блоковий шифр, обраний NIST у 2000 році (стандарт FIPS 197) як заміна старих схем типу DES; його внутрішня структура (раунди SubBytes → ShiftRows → MixColumns → AddRoundKey) забезпечує складну перестановково-замінну обробку 128-бітних блоків, а число раундів (10/12/14) залежить від довжини ключа (128/192/256 біт). У практичних системах AES рідко застосовують «в чистому вигляді»: його завжди вбудовують у конкретні режими роботи, які визначають властивості стійкості й поведінку при помилках — від категорично небезпечного для більшості задач ECB до широко вживаних AEAD-режимів (AES-GCM, AES-CCM), які одночасно забезпечують конфіденційність і автентифікацію даних. Для поточного шифрування або як будівельний блок для протоколів іноді використовують CTR-режим у парі з окремим MAC (CTR+HMAC) — підхід, що сьогодні вважається застарілим у порівнянні з AEAD-схемами, бо підвищує ризик помилкових композицій.

Критично важливий аспект проєктування — коректне управління параметрами режимів: для GCM стандарти рекомендують 96-бітовий попсе/IV і сувору гарантію його унікальності для кожної пари (ключ, потік), оскільки повторне використання попсе з тим самим ключем призводить до катастрофічного розкриття текстів; у дисковому шифруванні застосовується XTS-AES, оптимізований для блокових сховищ, але XTS не надає автентифікації, тож у випадках, де цілісність важлива, потрібно додатково проєктувати механізми перевірки. На рівні апаратної платформи AES має значні переваги: сучасні CPU підтримують інструкції AES-NI (Intel/AMD) або еквівалентні апаратні прискорювачі, що робить AES-GCM надзвичайно швидким і енергоефективним у серверних середовищах, тоді як на

пристроях без апаратного прискорення (мобільні, IoT) альтернативи на кшталт ChaCha20-Poly1305 можуть оказуватися продуктивнішими.

Реалізації AES підпадають під низку реалізаційно-орієнтованих ризиків: побічні канали (timing-атаки, cache-side-channel) можуть дозволити витік секретної інформації навіть при математично стійкому алгоритмі; неправильна генерація або повторне використання nonce/IV, помилки у паддінгу (padding-oracle) або «винайдення власного режиму» підвищують ймовірність компрометації. Тому в інженерній практиці вимагають constant-time реалізацій, ретельного тестування на побічні канали і використання перевірених бібліотек (наприклад, OpenSSL / BoringSSL, libsodium або апаратних модулів платформи), а також обмеження модифікацій криптографічних режимів лише до перевірених і стандартизованих.

Не менш важливе місце займає керування ключами: навіть найміцніший AES-ключ нічого не вартий без надійної генерації (висока ентропія), захищеного зберігання (HSM/KMS), політик ротації, розмежування доступу і чіткої аудиторної політики. У виробництві слід впроваджувати lifecycle-процеси для ключів — від генерації й розгортання до знищення — і застосовувати key-wrapping, полісі ретеншну та моніторинг доступу до криптографічних матеріалів. Інтеграція AES у протоколи (TLS, IPsec, SSH) також вимагає уваги до сумісності: обрані режими й параметри мають відповідати профілям реалізаторів і стандартам, інакше виникає ризик інтеперабельних помилок або слабких налаштувань.

На стандартному рівні організації повинні орієнтуватися на валідацію реалізацій (FIPS сертифікації там, де це потрібно), використовувати тестові вектори для перевірки коректності й регулярно проводити аудит конфігурацій. Моніторинг криптографічної підсистеми включає логування помилок шифрування/автентифікації, подій ротації ключів і аномалій у генерації nonce/IV; перед розгортанням змін у криптонастройках бажано виконувати інтеграційні тести в стейджинг-середовищі, щоб уникнути збоїв у робочому режимі.

У контексті майбутніх загроз слід враховувати квантовий вимір: симетричні алгоритми, включно з AES, піддаються прискоренню з боку алгоритму Гровера, що ефективно зменшує «еквівалентну» довжину ключа, тому для довготривалої конфіденційності рекомендовано використовувати більш довгі ключі (AES-256 має запас проти Grover-атак). Архітектурною практикою є криптоагільність — проектування систем так, щоб можна було плавно змінювати примітиви або параметри (ключова довжина, режим), і гібридні підходи там, де це доцільно.

Підсумовуючи, AES лишається базовим симетричним примітивом для захисту даних у каналі й у спокої завдяки збалансованій безпеці й продуктивності, але його безпечне застосування вимагає сукупності інженерних заходів: правильних режимів (перевага AEAD), строгого управління попсе/IV, апаратної або програмної оптимізації з урахуванням платформи, захисту реалізацій від побічних каналів, і надійного lifecycle-менеджменту ключів — тільки тоді AES забезпечить очікуваний рівень захисту в реальній інфраструктурі.

Принцип роботи AES

AES реалізує концепцію substitution–permutation network (мережа підстановок і перестановок): обробка кожного 128-бітного блоку даних виконується послідовністю раундів, що комбінують нелінійні підстановки (щоб забезпечити «заплутаність» — confusion) та лінійні перетворення (щоб домогтися «розсіювання» — diffusion). Перед першим раундом до вхідного блоку застосовується операція AddRoundKey (XOR з початковим раундовим ключем), далі йде цикл раундів: у кожному — SubBytes → ShiftRows → MixColumns → AddRoundKey, причому фінальний раунд опускає MixColumns. Число таких раундів залежить від довжини ключа: 10 для 128-бітного, 12 для 192-бітного і 14 для 256-бітного ключа, що підвищує стійкість при збільшенні ключа.

На рис. 5.2 наочно відображена поетапна обробка внутрішнього стану (state) 128-бітного блоку в класичній схемі AES: спочатку виконується початкове AddRoundKey, де стан XOR-иться з початковим шифрувальним ключем (cipher key), після чого слідує серія з дев'яти типових раундів, кожен із яких складається з чотирьох кроків — SubBytes → ShiftRows → MixColumns → AddRoundKey — і вбудовує у перетворення відповідний round key (на схемі показані матричні представлення підключів); фінальний, десятий раунд опускає операцію MixColumns і виконує лише SubBytes → ShiftRows → AddRoundKey, даючи на виході шифртекст. Схema підкреслює, що стан представлений у вигляді 4×4 байтової матриці, де SubBytes застосовує нелінійну S-box-підстановку до кожного байта, ShiftRows циклічно зрушує рядки для початку дифузії, а MixColumns лінійно змішує байти стовпців (матричне множення в полі $GF(2^8)$), посилюючи розсіювання впливу окремих бітів по всьому блоку; операція AddRoundKey на кожному кроці зв'язує поточний стан із відповідним раундовим ключем. Крім того діаграма ілюструє ідею key schedule — з основного ключа генерується послідовність раундових ключів (відображених як окремі матриці), що гарантує унікальність підключів для кожного раунду і ускладнює кореляції при криптоаналізі. Хоча на рисунку показано конфігурацію для AES-128 (10 раундів), та сама структура раунду застосовується і для AES-192/256, де

кількість основних раундів збільшується до 12 або 14 відповідно; зрозуміння цієї блок-схеми допомагає оцінити, чому поєднання підстановок, перестановок і раундових ключів забезпечує стійкість алгоритму, а також акцентує увагу на вимозі правильної реалізації (constant-time, коректне управління ключами та IV/nonce).

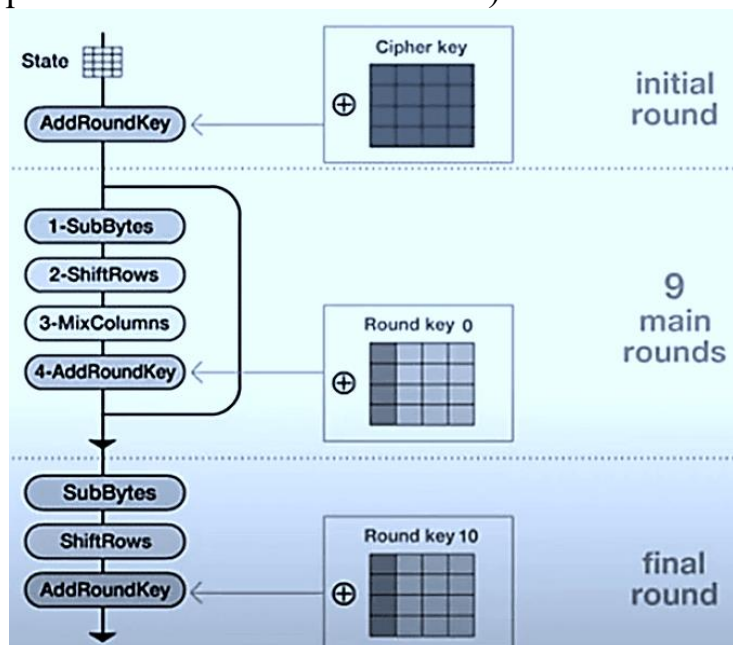


Рис. 5.2. Послідовність раундів AES (початковий раунд → основні раунди → фінальний раунд)

Джерело: URL: <https://www.netacad.com/courses/cybersecurity/>

Нелінійний шар SubBytes замінює кожен байт стану за допомогою S-box — заздалегідь заданої інверсійної таблиці на полі $GF(2^8)$ з додатковою афінною перетворювачем; S-box задуманий як бієктивна (зворотна) функція з високою нелінійністю, що зменшує ефективність лінійного й диференціального криптоаналізу. Операція ShiftRows зміщує байти в рядках матриці стану на різну кількість позицій, що перетворює локальну підстановку в міжбайтову залежність; MixColumns — це лінійне перетворення стовпців, яке виконує фіксоване матричне множення в полі $GF(2^8)$, рівномірно змішуючи байти і посилюючи дифузію: після кількох раундів будь-який біт вхідного блоку впливає на багато бітів виходу. AddRoundKey на кожному кроці просто XOR-ить поточний стан із відповідним раундовим ключем, тим самим пов'язуючи структуру перетворень із секретним ключем.

Генерація раундових ключів (key schedule) — ще один важливий компонент: з основного ключа розгортається послідовність підключів для

кожного раунду за допомогою операцій RotWord, SubWord та додавання констант Rcon; цей процес забезпечує те, що кожен раунд використовує інший підключ, ускладнюючи кореляції між раундами і підвищуючи стійкість до атак, що намагаються відновити ключ з часткової інформації. Хоча ключова схема проста щодо реалізації, її параметри були підібрані з урахуванням компромісу між безпекою і ефективністю.

На практиці AES не застосовують напряду до довгих потоків даних — для цього використовують режими роботи (CBC, CTR, GCM, XTS тощо), які визначають, як поєднувати блочні шифри для обробки довільних обсягів. У CTR-режимі AES шифрує послідовність лічильників (nonce||counter), результат яких XOR-иться з відкритим текстом, тим самим перетворюючи блоковий шифр на потоковий; у CBC кожен блок відкритого тексту XOR-иться з попереднім шифртекстовим блоком перед шифруванням, що дає залежність блоків; GCM комбінує CTR-режим для конфіденційності з алгоритмом GHASH для автентифікації (AEAD), що пояснює його широку популярність — але при цьому вимагає суворої унікальності nonce, оскільки повтор nonce призводить до повторення потоку і до можливості підробки. XTS-AES додає «tweak» (залежний від позиції блоку на диску) для шифрування блоків сховища, щоб ускладнити заміну блоків і позиційну атаку; проте XTS не забезпечує автентифікації, тому його застосування має супроводжуватись архітектурними міркуваннями щодо цілісності.

З теоретичної точки зору поєднання підстановок (S-box) і лінійних перетворень робить AES стійким до широкого класу математичних атак: субституційні таблиці забезпечують нелінійність, лінійні перетворення розподіляють вплив бітів по блоку, а раундова конструкція із незалежними раундовими ключами ускладнює зведення всього алгоритму до однієї простої оберненої операції. Однак на практиці безпека залежить не лише від математичної конструкції, а й від коректної реалізації: реалізації мають бути constant-time, уникати виразних таблиць доступу (table lookups) на платформі з кеш-каналами, і правильно поводитись з IV/nonce/ключами.

На наведеній схемі (рис. 5.3) зображено послідовну обробку одного 128-бітного блоку в алгоритмі AES-128: спочатку відкритий текст P піддається операції AddRoundKey з початковим раундовим ключем K_0 , а далі починається цикл раундів, у кожному з яких послідовно виконуються перетворення SubBytes $\rightarrow$ ShiftRows $\rightarrow$ MixColumns $\rightarrow$ AddRoundKey з відповідним раундовим ключем ($K_1 \dots K_9$). Операція SubBytes замінює кожний байт стану через нелінійну S-box, забезпечуючи криптографічну «заплутаність», ShiftRows зсуває байти в рядках матриці стану і починає процес дифузії, а MixColumns лінійно змішує байти стовпців (матричне

множення в полі $GF(2^8)$), що розсіює вплив окремих бітів по всьому блоку; на кожному кроці `AddRoundKey` вбудовує секретність за допомогою XOR з раундовим ключем. Фінальний, десятый раунд відрізняється тим, що в ньому опущено крок `MixColumns` — виконується лише `SubBytes` → `ShiftRows` → `AddRoundKey` з ключем K_{10} , після чого отримується шифртекст C ; ця конструкція знижує симетричність перетворень і дозволяє отримати зворотну функцію при розшифруванні. Раундові ключі $K_0 \dots K_{10}$ не є довільними: вони генеруються з основного секретного ключа за допомогою `key-schedule` (операцій `RotWord`, `SubWord` та додавання констант `Rcon`), що гарантує різні підключі для кожного раунду і ускладнює кореляції між раундами.

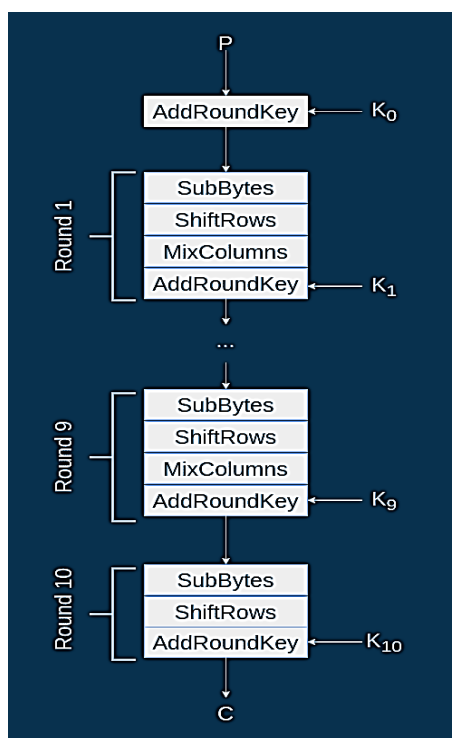


Рис. 5.3. Схема раундів AES-128 (`SubBytes` → `ShiftRows` → `MixColumns` → `AddRoundKey`)

Джерело: URL: <https://www.netacad.com/courses/cybersecurity/>

Хоча на схемі показано AES-128 з десятима раундами, для ключів довжини 192 та 256 біт число раундів відповідно збільшується до 12 і 14, при цьому внутрішня структура кожного раунду лишається незмінною. У сукупності така архітектура—мережа підстановок і перестановок (`substitution-permutation network`)—поєднує нелінійні й лінійні шари з унікальними раундовими ключами, що створює високу стійкість до лінійного й диференціального криптоаналізу; проте реальна безпека також

залежить від правильної реалізації (constant-time), коректного управління ключами і унікальності IV/nonce у вибраних режимах роботи.

Отже, коротко: принцип AES — це початкове AddRoundKey, серія раундів substitution → permutation → linear mixing → AddRoundKey з різними раундовими ключами, і фінальне AddRoundKey; ключовим забезпеченням стійкості є S-box-нелінійність, MixColumns-дифузія і надійний key schedule. Розуміння цього принципу допомагає проектувати безпечні режими використання (з унікальними nonce, AEAD-підходами та належним управлінням ключами) і формувати вимоги до реалізації (constant-time, апаратне прискорення, сертифікація), щоб AES у поєднанні з правильною інженерією дійсно забезпечував очікуваний рівень захисту.

ChaCha20

ChaCha20 — це поточний/стрімовий шифр, який у парі із Poly1305 як MAC утворює широко використовуваний AEAD-фреймворк ChaCha20-Poly1305. Його сильні сторони — висока швидкість у програмному виконанні на платформах без апаратного AES-прискорення (мобільні пристрої, IoT), відносна простота імплементації з кращою стійкістю до побічних каналів типу timing-атак, та стандартний, надійний набір RFC-параметрів (знову-таки рекомендований 96-бітовий nonce у сучасних застосуваннях). У практиці ChaCha20-Poly1305 часто застосовують там, де AES-NI відсутній, або як альтернативу в TLS (його підтримують у TLS 1.2/1.3), а також у VPN-рішеннях наступного покоління (наприклад, WireGuard використовує ChaCha20-Poly1305 як основний AEAD). Для мобільних і вбудованих систем ChaCha20 часто є кращим вибором через баланс продуктивності і безпеки.

ChaCha20 — це сучасний потоковий (stream) шифр, спроектований Даніелем Бернштейном як швидка, проста й безпечніша у реалізації альтернатива для платформ без апаратного прискорення AES. Його принцип роботи базується на детермінованому генеруванні довгого псевдовипадкового «keystream» (потoku байтів), який потім XOR-иться з відкритим текстом для отримання шифртексту (розшифрування — та сама операція XOR із тим самим потоком). Це дає властивості потокового шифру: відсутність падінгу, можливість довільного доступу до позицій у потоці за допомогою лічильника (counter) та просту обробку довгих/коротких повідомлень.

Технічно ChaCha20 працює зі станом із 16 слів по 32 біти (всього 512 біт), який складається з чотирьох констант (fixed constants), восьми слів ключа (256-бітний ключ), одного 32-бітного або 64-бітного лічильника (counter) і трьох або трьох-/чотирьох слів nonce/IV залежно від специфікації.

Для генерації keystream-блока алгоритм виконує послідовність «раундів», кожен з яких складається з так званих quarter-round перетворень — простих операцій додавання по модулю 2^{32} , XOR і циклічного лівого зсуву (rotate). У стандартній версії ChaCha20 застосовують 20 раундів (поєднання колонних і діагональних четверть-раундів), що дає високу дифузію й нелінійність на практиці. Після завершення 20 раундів до отриманого стану додають початковий стан (feed-forward), отримують 16 слів вихідного блоку і серіалізують їх у байти — це один блок keystream, який XOR-иться з 64-байтовим блоком відкритого тексту.

У парі з Poly1305 цей шифр утворює AEAD-конструкцію ChaCha20-Poly1305. У цій схемі з ChaCha20 спочатку генерується спеціальний одноразовий ключ для Poly1305 (часто шляхом запуску ChaCha20 з counter = 0), потім повідомлення шифрується ChaCha20, а тег автентичності (MAC) обчислюється алгоритмом Poly1305 над додатковими даними (AAD), шифртекстом і їх довжинами у визначеному порядку. При розшифруванні спочатку перевіряють тег; тільки після успішної валідації виконується відновлення відкритого тексту — це запобігає атакам, що модифікують шифртекст.

Ключові практичні властивості та наслідки принципу роботи:

- Унікальний nonce/IV обов'язковий. Для безпеки необхідно гарантувати, що пара (ключ, nonce) не повторюється; повтор nonce призводить до повтору keystream і до можливості відновлення XOR-різниці двох повідомлень (катастрофічна вразливість). IETF-версія рекомендує 96-бітовий nonce із 32-бітовим counter-полем (RFC 8439).
- Висока швидкість в програмному виконанні. Через простоту операцій (додавання, XOR, оберти) ChaCha20 дуже швидкий на CPU без AES-прискорення і менш чутливий до timing-каналів у порівнянні з таблицями S-box в реалізаціях AES.
- Поточкова природа. Дає можливість випадкового доступу до будь-якої «позиції» у потокові за допомогою зміни лічильника, що зручно для деяких застосувань (але слід бути обережним із nonce/керуванням позиціями).
- AEAD через Poly1305. Поєднання шифрування і автентифікації дає захист від підробок (forge) і необхідно для безпечної практики: ніколи не використовувати ChaCha20 без автентифікації повідомлення.

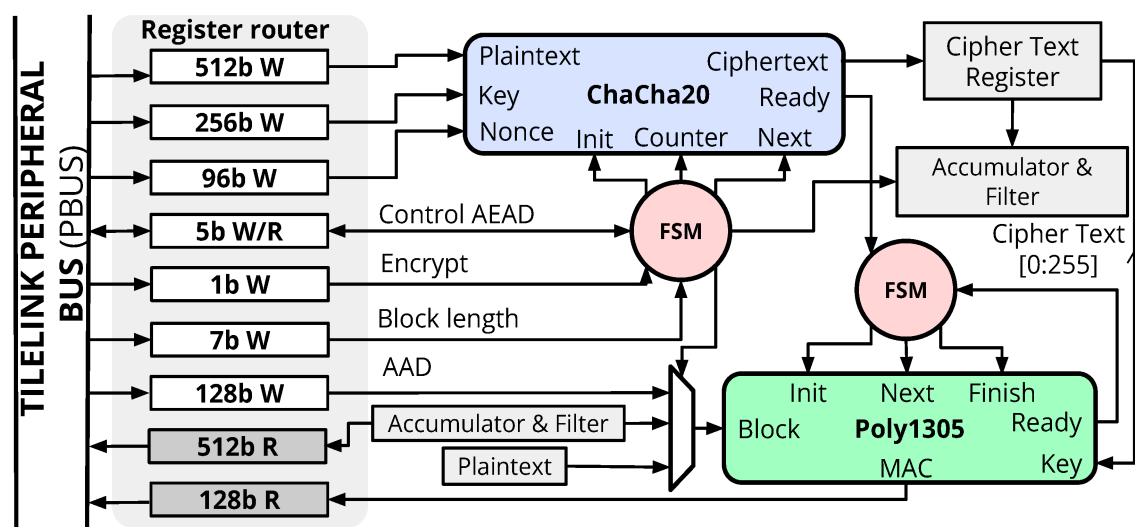


Рис. 5.4. Блок-діаграма апаратної реалізації ChaCha20-Poly1305 (інтерфейс і потік даних)

Джерело: URL: <https://www.netacad.com/courses/cybersecurity/>

На схемі (рис. 5.4) показано апаратну (hardware) реалізацію AEAD-двійки ChaCha20-Poly1305 із регістровим інтерфейсом до шини периферії (TileLink/PBus) та керуючими скінченними автоматами. Зліва — набір регістрів керування й даних (ключ, nonce, AAD, довжина блока, прапори керування) з різними ширинами шини (128/256/512 біт тощо), куди процесор або DMA записує параметри й вхідні дані; ці регістри приписані до входів ядра ChaCha20 і до блока підготовки Poly1305. Посередині — блок ChaCha20, що ініціалізується константами/ключем/nonce і інкрементним лічильником; він генерує keystream-блоки (через 20 раундів) і подає їх на операцію XOR з відкритим текстом, результатом чого є шифртекст. Одночасно шифртекст і AAD надходять у блок Poly1305: спочатку ChaCha20 (counter=0) видає одноразовий ключ для Poly1305, потім Poly1305 накопичує AAD||ciphertext||lengths і обчислює 16-байтовий тег автентичності. Два FSM (контролери) координують послідовність операцій — ініціалізацію, генерацію наступного блока, передачу даних до кеш-регістрів і завершення; додаткові блоки «Accumulator & Filter» вирівнюють ширини інтерфейсів і формують потоки 128/512 біт для шини та для MAC. На виході є регістр шифртексту для зчитування периферією та регістр тега; примітка на схемі підкреслює критичні вимоги: коректна ініціалізація ключа/nonce, послідовна перевірка тегу і синхронізація FSM для уникнення повторного використання (key, nonce).

Реалізаційні та експлуатаційні рекомендації: використовувати перевірені бібліотеки (наприклад, libsodium, BoringSSL, OpenSSL, NaCl), суворо слідкувати за політиками генерації й ротації ключів, забезпечити

ненаповнюваність (unique) nonce для кожного ключа й не реалізовувати власноручні оптимізації, які можуть призвести до повторного використання nonce. При роботі з ChaCha20-Poly1305 перевірка теги автентичності має виконуватись до обробки даних шифртексту; також варто застосовувати constant-time порівняння тегів для уникнення побічних атак. Нарешті, ChaCha20 добре підходить для мобільних і IoT-платформ, для VPN (наприклад, WireGuard) і в TLS як альтернатива AES-GCM там, де відсутнє апаратне прискорення — але правило унікального (ключ, nonce) універсальне для всіх застосувань.

Twofish

Twofish — блоковий шифр із 128-бітним блоком і ключами до 256 біт, який був фіналістом конкурсу AES. Хоча він не став стандартом, Twofish зберігає сильну криптографічну репутацію і використовується в деяких проєктах для дискового шифрування або як опція в системах, що прагнуть криптоагільності або сумісності з історичними рішеннями (прикладом можуть бути деякі реалізації шифрування у засобах типу VeraCrypt). Twofish має складнішу внутрішню структуру, ніж ChaCha20, і не отримує такого універсального апаратного прискорення, як AES, тому його застосування зараз частіше обумовлене вимогами сумісності або політичною/архітектурною вибірковістю, а не абсолютною продуктивністю.

Twofish — це 128-бітовий блоковий шифр (підтримує ключі 128/192/256 біт), спроектований як швидка, гнучка й криптографічно стійка альтернатива, і його принцип роботи поєднує класичну структуру Feistel-мережі з низкою додаткових «сміливих» інженерних рішень (key-dependent S-box'и, MDS-матриця, PHT, початкова/фінальна «whitening»). Вхідний 128-бітний блок представлено як чотири 32-бітні слова; перед основними раундами застосовується input whitening — XOR з кількома підключами, а після завершення всіх раундів — output whitening, що ускладнює пряму атаку на простий Файстел. Основний процес — 16 раундів Feistel: у кожному раунді праві слова обробляються через функцію FFF (що включає ключ-залежні S-box'и, MDS-комбінацію і PHT), результати якої комбінуються з лівими словами через операції додавання/XOR і циклічні зрушення, після чого пари слів міняються місцями (типова Feistel-логіка).

Ключова відмінність Twofish — ключ-залежні S-box'и: замість фіксованих підстановок S-box'и конструюються з бітів секретного ключа (через послідовність фіксованих q-перестановок і лінійних перетворень), що ускладнює побудову загальних атак і робить криптоаналіз залежним від конкретного ключа. Після S-box'ів байти кожного 32-бітного слова

проходять через MDS (maximum distance separable) матрицю у полі $GF(2^8)$ — це забезпечує сильну локальну дифузію: невелика зміна в одному байті швидко впливає на весь вихід слова. Усередині функції FFF також використовується Pseudo-Hadamard Transform (PHT) для комбінації двох 32-бітних значень і підсилення нелінійності та розподілу бітів перед тим, як результат буде включено в раундові перетворення. Генерація підключів (key schedule) — окрема, важлива частина: з основного ключа будується набір S-box-ключів (через RS-матрицю/лінійне кодування) і 40 підключів (2 підключі на раунд $\times 16 = 32$ плюс 8 для whitening) за допомогою h-функції, що використовує ті самі ключ-залежні S-box'и; це гарантує, що і структура S-box'ів, і підключі тісно пов'язані з секретом.

З точки зору безпеки Twofish був ретельно проаналізований (був фіналістом AES-конкурсу) і не має відомих практично ефективних атак проти повних 16 раундів при коректній параметризації; проте, як і будь-який алгоритм, його реальна стійкість залежить від коректної реалізації — захисту від побічних каналів (constant-time, уникнення таблиць у кеш-вразливих середовищах), надійного управління ключами і недопущення повторного використання підключів/IV у відповідних режимах. У практиці Twofish застосовують там, де важлива криптоагільність або сумісність (наприклад, де проєкт вимагає альтернативи AES або підтримки історичних режимів, як у деяких реалізаціях VeraCrypt), але на сучасних платформах із підтримкою AES-NI зазвичай обирають AES через апаратне прискорення. Для інженерної реалізації рекомендовано використовувати перевірені бібліотеки/референс-реалізації, виконувати constant-time оптимізації і тестувати на побічні канали; при цьому Twofish є вільним від патентних обмежень і може використовуватись без ліцензійних бар'єрів.

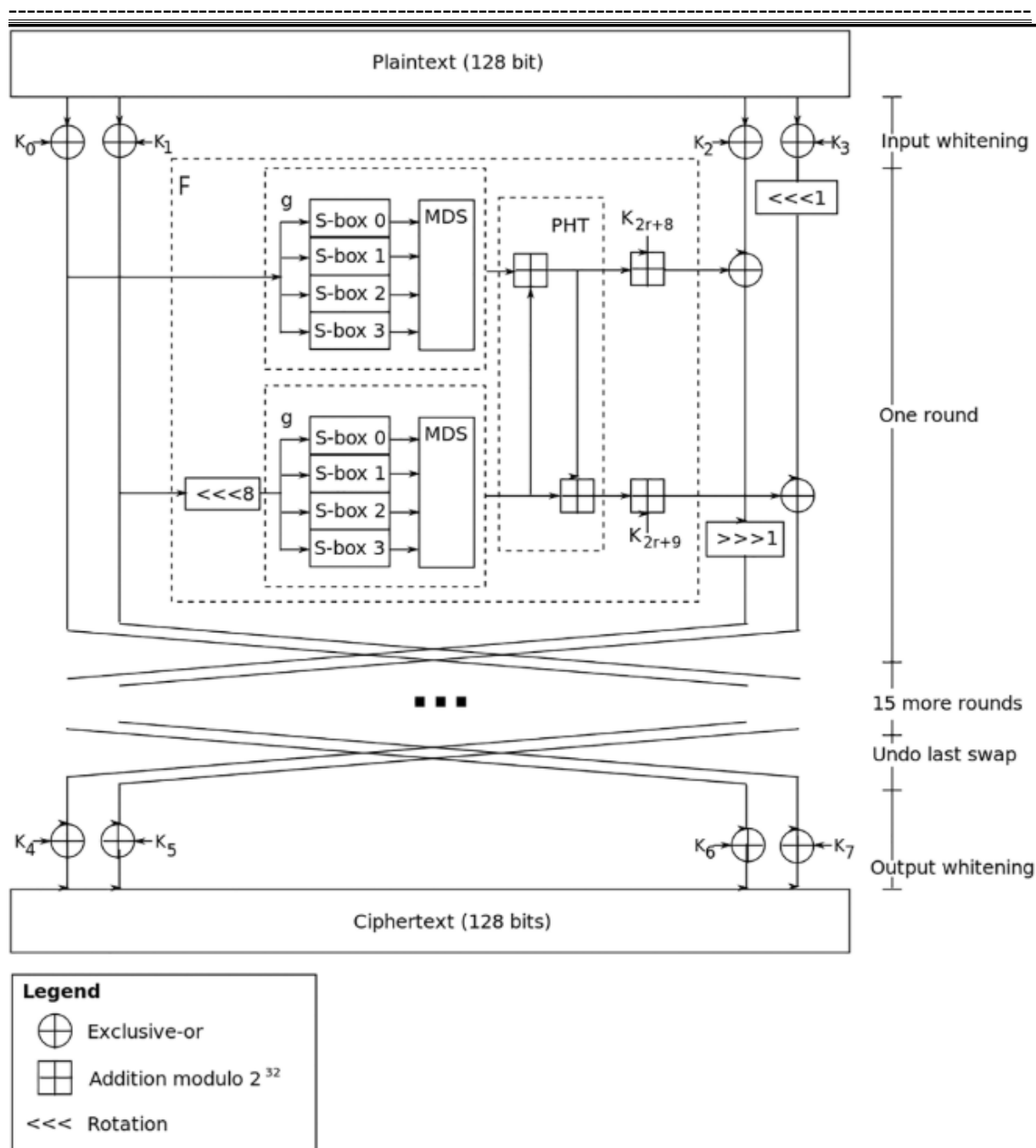


Рис. 5.5. Детальна блок-схема раунду Twofish — g-блоки, MDS, PHT та раундовий вихід

Джерело: URL: <https://www.netacad.com/courses/cybersecurity/>

На рис. 5.5 демонструється внутрішня архітектура одного Feistel-раунду алгоритму Twofish і послідовність перетворень, що приводять до раундового виходу. Вхідний 128-бітний блок представлений як чотири 32-бітні слова; перед основними раундами на них застосовано input-whitening, а після завершення всіх раундів — output-whitening (це показано стрілками

в початковій і кінцевій частинах схеми). Серце раунду — два незалежні g-блоки, кожний з яких приймає 32-бітне слово і обробляє його через чотири key-dependent S-box'и (S-box'и будуються з бітів секретного ключа), далі результати проходять через MDS-матрицю (maximum distance separable) для сильної локальної дифузії, що забезпечує швидке розповсюдження змін одного байта на весь вихід g-блоку. Виходи двох g-блоків позначені як T0 і T1; ці значення комбінуються через Pseudo-Hadamard Transform (PHT) — операцію, що поєднує два 32-бітні слова у дві нові величини з підсиленою нелінійністю і розподілом бітів — після чого до них додаються відповідні раундові підключі (round keys) і результат XOR'ється/додається до лівих слів у стандартній логіці Feistel; наприкінці раунду зазвичай відбувається обмін місцями (swar) половин, як в типовій Feistel-мережі.

На схемі також позначено зв'язок із key-schedule: з основного секретного ключа генерується набір ключових матеріалів, які використовуються і для побудови ключ-залежних S-box'ів, і як раундові підключі (включно з підключами для whitening). Така тісна залежність структури S-box'ів і підключів від ключа підвищує стійкість до загальних (key-independent) атак і ускладнює криптоаналіз. Ілюстрація корисна для розуміння, чому Twofish поєднує властивості Feistel-мережі з потужними лінійними перетвореннями (MDS) і ключ-залежною нелінійністю (S-box), а також нагадує про реалізаційні вимоги: захист від побічних каналів, correct-time реалізації S-box'ів і надійне управління ключами/IV, бо безпечна математика втрачає силу при некоректній інженерії.

Що спільного й що відрізняє алгоритми у практиці застосування?

По-перше, сучасні рекомендації категорично наполягають на використанні автентифікованого шифрування (AEAD) — тобто не лише шифрування байтів, а й одночасної перевірки їхньої цілісності — тому в більшості сценаріїв пріоритетні AES-GCM або ChaCha20-Poly1305. По-друге, вибір між AES і ChaCha20 часто зводиться до апаратної платформи: там, де доступне AES-NI, AES-GCM даватиме кращу продуктивність; там, де апаратного прискорення немає, ChaCha20-Poly1305 може бути швидшим і безпечнішим щодо імплементаційних ризиків. Twofish залишається опцією для криптоагільних систем або специфічних застосунків, але в більшості сучасних стеків він зустрічається рідше.

Практичні застосування симетричних алгоритмів включають: захист каналів зв'язку (TLS/SSH/IPsec — AEAD-спеки), шифрування даних у спокої (дискові контейнери, файлові системи — AES-XTS або AEAD-рушії на рівні файлової системи), захист баз даних і резервних копій (включно з ключовим менеджментом і обгортанням ключів), VPN і тунелювання (де

часто вибір між AES і ChaCha20 залежить від платформи), а також вбудовані системи/IoT (де ChaCha20 має переваги). Ключовим доповненням до будь-якого вибору алгоритму є надійне керування ключами: генерація з достатньою ентропією, безпечне зберігання (KMS/HSM), регулярна ротація, політика ретеншну і контролю доступу; без цього навіть найміцніший алгоритм не забезпечить безпеки.

Нарешті — типові помилки й практичні застереження: не слід винаходити власні режими чи поєднання шифрів; для AEAD-режимів необхідно суворо уникати повторного використання nonce/IV з одним і тим самим ключем (особливо для GCM це критично); у режимах CBC потрібно уникати вразливостей padding-oracle; імплементації мають бути стійкі до побічних каналів (constant-time, захист від cache-timing); і завжди краще покладатися на перевірені бібліотеки (OpenSSL/BoringSSL, libsodium, BoringCrypto, модулі платформи) ніж на самописну криптографію. Для довготривалого збереження конфіденційності варто також враховувати постквантові ризики (для симетричних схем достатньо підвищити параметри — наприклад, AES-256 дає запас щодо квантових атак), а в архітектурі передбачати криптоагільність, щоб можна було замінити примітиви без кардинальних змін.

У підсумку: AES — універсальний стандарт із сильним апаратним прискоренням для серверних і дискових сценаріїв; ChaCha20-Poly1305 — ефективний і безпечний вибір для мобільних, IoT і програмних реалізацій; Twofish — надійна альтернатива для специфічних випадків і для систем, що прагнуть криптоагільності. Правильний вибір та безпечна реалізація (AEAD, неповторні nonce, керування ключами, перевірені бібліотеки) — ось що визначає реальну ефективність симетричного шифрування в практичних завданнях.

5.3. АСИМЕТРИЧНІ АЛГОРИТМИ (RSA, ECC, ПОСТКВАНТОВІ МЕТОДИ) У ЗАХИСТІ КОМУНІКАЦІЙ

Асиметричні алгоритми — це сімейство криптографічних примітивів, у яких використовуються пари ключів (публічний / приватний) для безпечного обміну, автентифікації й цифрового підпису; у контексті захисту комунікацій найбільш відомі представники — RSA і ECC (еліптичні криві), а сьогодні дедалі більше уваги приділяють постквантовим (PQC) методам, стійким до атак із використанням квантових комп'ютерів. У своїй сукупності ці алгоритми визначають, як встановлюються сесійні ключі, як перевіряється автентичність кінцевих точок і як гарантується неможливість

відмови від підписаних повідомлень — тобто вони задають довірчий шар усіх сучасних протоколів зв'язку.

RSA базується на задачі факторизації великих чисел: публічний ключ дається (n, e) , приватний — (d) , і без знання d дешифрування або підробка підпису неможливі при умовній складності факторизації. ECC замінює арифметику на операції над точками еліптичних кривих: при тих самих рівнях безпеки ECC дає значно коротші ключі й підписи (менше трафіку, менше затримки), що робить її привабливою для мобільних і вбудованих систем. Практично RSA/ECC виконують дві ролі в захисті комунікацій: (1) обмін/встановлення симетричних сесійних ключів (наприклад, через DH/ECDH або як частина TLS handshake), і (2) автентифікація / цифрові підписи (сертифікати X.509, підписи повідомлень), які запобігають man-in-the-middle атакам і гарантують цілісність та походження даних.

У практичних протоколах (TLS, SSH, S/MIME, IPsec) масово застосовується гібридна архітектура: асиметрія (RSA, ECDH) використовується лише для безпечного ідентифікованого встановлення сесії, після чого для передачі даних застосовуються симетричні AEAD-алгоритми (AES-GCM, ChaCha20-Poly1305) через більш ефективні ключі. Важливе доповнення — ephemeral (тимчасові) ключі, наприклад ECDHE у TLS, які надають forward secrecy: навіть якщо приватний ключ сервера буде скомпрометовано пізніше, минулі сесії, зашифровані з використанням одноразових ECDHE-сесій, залишаться недешифрованими.

Постквантова криптографія додає третій рівень у цю картину. Квантові алгоритми (алгоритм Шора) загрожують класичним асиметричним схемам RSA та ECC, тому в архітектурному плані вже зараз варто планувати міграцію: криптоагільність системи (можливість змінювати алгоритми без переробки всього стеку), тестування гібридних KEX (одночасне використання класичного + PQC KEX для захисту від “collect-now-decrypt-later”), і поетапне впровадження стандартизованих PQC-алгоритмів у протоколи й PKI. Це означає не лише заміну алгоритмів, а й оцінку продуктивності, розмірів ключів/підписів, сумісності з існуючими сертифікаційними системами та вимогами до зберігання й ретеншну даних.

Операційні аспекти — не менш важливі: асиметричні ключі потрібно генерувати з надійної ентропії, зберігати у захищених сховищах (HSM/KMS), контролювати доступ і ротацію сертифікатів; PKI-процеси (видача, оновлення, відкликання сертифікатів) мають бути описані в політиках і автоматизовані (ACME/Let's Encrypt-подібні механізми для доступних сервісів). Для підписів коду і сертифікації варто використовувати апаратні модулі та практики, що знижують ризик крадіжки приватного ключа (мульти-підписи, threshold crypto, HSM-backed signing).

З точки зору безпеки реалізації — важливо уникати класичних помилок: не використовувати застарілі параметри (короткі ключі, слабкі криві), застосовувати перевірені криптобібліотеки (OpenSSL/BoringSSL, libsodium, модулярні PQC-реалізації), захищати операції від побічних каналів (constant-time реалізації, захист від timing/cache attacks) і проводити регулярні аудити / Fuzz-тестування. Також варто впроваджувати моніторинг і логування аномалій аутентифікації та підозрілих handshake'ів, щоб оперативно виявляти спроби компрометації.

Узагальнюючи: асиметричні алгоритми — це фундамент довіри в мережових комунікаціях: вони забезпечують встановлення секретних сесій, автентифікацію й підписи; вибір між RSA і ECC визначається компромісом продуктивності й розмірів ключів, а постквантові методи — це наступний крок у забезпеченні довготривалої конфіденційності. Практична стратегія — діяти зараз у напрямку криптоагільності: застосовувати сучасні, рекомендовані параметри (ephemeral ECDHE для forward secrecy), інтегрувати PQC у гібридних конфігураціях для чутливих сценаріїв, керувати ключами через HSM/KMS і ретельно тестувати сумісність та продуктивність перед масштабним розгортанням.

RSA (RIVEST–SHAMIR–ADLEMAN)

Шифрування з асиметричним ключем — цей метод використовує пару ключів: публічний (для шифрування) і приватний (для дешифрування). Протоколи, такі як RSA, використовують цю модель для забезпечення безпеки в Інтернеті, зокрема для електронного підпису та захисту електронної пошти. Асиметричне шифрування дозволяє уникнути проблем з безпечним обміном ключами, оскільки публічний ключ можна передавати відкрито, тоді як приватний залишається в секреті. Однак цей метод є більш повільним порівняно з симетричним шифруванням, тому часто використовується у поєднанні з симетричними алгоритмами для шифрування даних. У таких випадках асиметричне шифрування застосовується для захисту ключа симетричного алгоритму, який потім використовується для шифрування великого обсягу даних. Крім RSA, існують і інші алгоритми асиметричного шифрування, такі як ECC (Elliptic Curve Cryptography), які забезпечують більшу безпеку при меншій довжині ключів.

Криптосистема RSA — одна з найпоширеніших асиметричних схем, яку застосовують для захисту конфіденційності, підтвердження джерела та збереження цілісності даних. Вона була запропонована 1977 року Р. Рівестом, А. Шаміром і Л. Адлеманом і названа за ініціалами їхніх прізвищ. У RSA використовуються дві різні ключові величини: публічний ключ, відкрито доступний і призначений для шифрування, та приватний ключ, що

зберігається у секреті й потрібен для розшифрування. Безпека методу спирається на те, що поділ великого числа на прості множники є складною обчислювальною задачею — це робить можливим просте створення шифртексту за допомогою публічного ключа, але практично неможливим його відновлення без відповідного приватного ключа.

Шифрування повідомлення відбувається за допомогою відкритого ключа. Повідомлення перетворюється на число, яке є меншим за n , і шифрується за допомогою формули $C = M^e \bmod n$, де M — це оригінальне повідомлення, а C — зашифроване повідомлення. Для розшифрування використовують закритий ключ, застосовуючи формулу $M = C^d \bmod n$, що дозволяє відновити початкове повідомлення. Важливою властивістю RSA є те, що зашифроване повідомлення неможливо розшифрувати без доступу до закритого ключа, що забезпечує високий рівень безпеки.

RSA володіє низкою сильних сторін: її безпека базується на труднощі факторизації великих цілих чисел, що робить практичний злом без знання приватного ключа надзвичайно складним; крім того, відсутність потреби попередньо обмінюватися секретними ключами спрощує встановлення захищеного зв'язку між сторонами. Водночас у RSA є й істотні обмеження: операції з дуже великими числами потребують значних обчислювальних ресурсів, через що схема порівняно повільна для шифрування великих обсягів даних. Через це RSA зазвичай застосовують для захисту невеликих повідомлень або — найчастіше — для безпечного обміну симетричними ключами, після чого самі обміни даними захищають швидкими симетричними алгоритмами.

Практично RSA лежить в основі багатьох протоколів і сервісів безпеки (наприклад, TLS/HTTPS і механізми цифрових підписів), а також широко використовується в електронній комерції, платіжних системах і в інших секторах, де потрібна автентифікація та конфіденційність. Разом з тим поява квантових обчислювальних технологій ставить перед RSA серйозний виклик: квантові алгоритми теоретично можуть значно зменшити складність факторизації й зробити такі схеми вразливими. Попри це наразі RSA залишається одним із найбільш поширених і надійних засобів криптографічного захисту, поки відбувається робота над міграцією до постквантових рішень.

ЕСС (ЕЛІПТИЧНІ КРИВІ)

ЕСС переносить основну складність із факторизації на проблему дискретного логарифма в групі точок еліптичної кривої (ECDLP), що дає

значно коротші ключі при порівнянній безпеці: наприклад, ключі 256 біт у ECC забезпечують безпеку на рівні приблизно 3072-бітного RSA, тому ECC широко застосовують для ECDH/ECDHE (обмін сесійними ключами) і для підписів (ECDSA, EdDSA). На практиці вибір кривої і параметрів критично важливий: сучасні рекомендації віддають перевагу добре вивченим кривим або сучасним варіантам типу Curve25519/Ed25519, котрі мінімізують ризики параметризації і спрощують secure implementation; у протоколах рекомендують ephemeral-ключі (ECDHE) для забезпечення forward secrecy, а також constant-time реалізації для захисту від побічних каналів. ECC дає переваги в продуктивності і зменшеному мережевому навантаженні, але так само під загрозою квантових атак, тому на етапі архітектурного планування варто передбачати криптоагільність: можливість додати PQC або перевести KEX у гібридний режим без кардинального перероблення стеку.

Криптографія на еліптичних кривих (ECC) — це клас асиметричних схем, які замість класичних груп чисел використовують арифметику точок на еліптичній кривій над скінченним полем, і саме ця математична структура забезпечує їм високу криптографічну стійкість при значно компактніших параметрах. Безпека ECC базується на складності задачі дискретного логарифма в групі точок кривої (ECDLP): операції множення точки на скаляр виконуються ефективно, тоді як обернена дія — відновлення скаляра за двома точками — залишається практично нерозв'язною для поточних алгоритмів. Ця властивість дозволяє отримувати рівень захисту, порівнянний із RSA, але при значно менших розмірах ключів — що зменшує витрати пам'яті, мережевого трафіку й обчислювальні затрати, а отже робить ECC особливо привабливою для мобільних пристроїв, вбудованих систем і розподілених застосунків.

На практиці еліптичні криві застосовують у механізмах встановлення ключів (ECDH/ECDHE), у цифрових підписах (ECDSA, EdDSA) і в протоколах аутентифікації — їх підтримують TLS/SSL, SSH, VPN-рішення та багато сучасних платіжних і блокчейн-систем. Водночас вибір конкретної кривої та її параметрів вимагає уважного підходу: частина кривих має історію дискусій щодо безпечної параметризації, тому сьогодні рекомендують перевірені профілі (наприклад, Curve25519/Ed25519 або ретельно верифіковані NIST-криві) і уніфіковані стандарти (NIST, SECG тощо). Реалізація ECC також потребує захисту від побічних каналів і constant-time операцій, інакше математично стійкий примітив може стати вразливим через інженерні помилки.

Підсумовуючи, ECC дає значну економію ресурсів при збереженні високого рівня захисту, що робить її ключовою технологією для сучасних мережевих протоколів, мобільних застосунків і блокчейн-інфраструктур;

одночасно необхідно ретельно обирати криві, слідувати рекомендованим стандартам і впроваджувати надійні інженерні практики для забезпечення реальної, а не тільки теоретичної, безпеки.

Принцип криптосистем на основі еліптичних кривих (Elliptic Curve Cryptography, ECC) базується на використанні математичних властивостей еліптичних кривих для створення ефективних і безпечних криптографічних алгоритмів. Еліптичні криві є геометричними об'єктами, які визначаються рівнянням типу $y^2 = x^3 + ax + b$, де a та b — це параметри, що задають конкретну криву, а x і y — змінні, що представляють точки на кривій.

Суть застосування ECC полягає в тому, що обчислення дискретного логарифма в групі точок на еліптичній кривій є на практиці надзвичайно важким завданням, і саме це забезпечує криптографічну міцність схем: приватний ключ — випадковий скаляр, а публічний ключ отримують множенням фіксованої генераторної точки кривої на цей скаляр, тому, навіть знаючи публічний ключ, зворотне відновлення приватного скаляра вважається обчислювально не виправданим. Головна перевага ECC — можливість досягти того самого рівня безпеки, що й у традиційних підходах (наприклад, RSA), при набагато коротших ключах, через що зменшуються вимоги до зберігання, пропускну здатності й обчислювальних ресурсів; ці властивості роблять ECC привабливою для мобільних і вбудованих систем. На практиці реалізації спираються на стандартизовані, ретельно відібрані криві (наприклад, NIST P-256, P-384 або сучасні варіанти на кшталт Curve25519), що допомагає знизити ризики, пов'язані з параметризацією, і забезпечити інтеоперабельність; ECC широко використовується для встановлення сесій (ECDH/ECDHE), у цифрових підписах (ECDSA, EdDSA) і в протоколах захищених з'єднань (TLS/SSL), а також у блокчейн-середовищах і системах електронних платежів для ефективного підписування й управління ключами.

Постквантові методи (PQC)

Постквантова криптографія — це клас алгоритмів, спроектованих на математичних припущеннях, для яких не відомі ефективні квантові атаки (тобто вони повинні витримати теоретичні атаки на квантових машинах), і включає кілька сімейств із різними компромісами: решіткові схеми (LWE/NTRU та ін.) обіцяють гнучкість і невисоку вартість у багатьох сценаріях; кодово-орієнтовані (наприклад, McEliece) мають довгу історію стійкості, але великі ключі; hash-based підписи (SPHINCS+) дають прості математичні гарантії для підписів; існують також мультиваріантні та інші конструкції. Практична інтеграція PQC вимагає оцінки продуктивності, розмірів ключів/підписів і сумісності з існуючими протоколами; тому

сьогодні рекомендована стратегія — поступова міграція через криптоагільні архітектури і гібридні КЕХ (одночасне застосування класичного алгоритму та PQC-алгоритму в одному handshake), що забезпечує захист від «collect-now-decrypt-later». Для впровадження PQC потрібні також оновлені політики щодо зберігання ключів (HSM/KMS повинні підтримувати нові формати), тестування на вразливості реалізацій і планування операційних процесів (видача/відкликання сертифікатів, ретеншн). Незважаючи на те, що стандартизація і реалізації PQC проходять фазу валідації (наприклад, процес NIST), практичні організації повинні починати тестування і впровадження гібридних рішень уже зараз, щоб зменшити ризики при появі потужних квантових обчислювальних ресурсів.

5.4. ХЕШ-ФУНКЦІЇ, ЦИФРОВІ ПІДПИСИ І КОНТРОЛЬ ЦІЛІСНОСТІ ІНФОРМАЦІЇ

Хеш-функція — це алгоритм, який перетворює довільний блок даних у фіксованої довжини відбиток (хеш-значення). Мета такого перетворення — отримати компактний і однозначний «відбиток» вхідної інформації: невелика зміна у вихідних даних призведе до радикальної зміни хешу, тож за допомогою порівняння відбитків легко виявити будь-яку підміну або пошкодження.

Криптографічні хеш-функції мають кілька ключових властивостей: вони повинні бути «односпрямованими» (неможливо відновити початкові дані за хешем), стійкими до першої і другої області обернення (preimage/second-preimage resistance) та малоймовірними до колізій (дві різні входи дають один і той же хеш). Також очікують ефекту «лавини»: невелика зміна вхідних бітів призводить до великої, непередбачуваної зміни вихідного значення.

Практичні застосування хеш-функцій дуже широкі. Найчастіше їх використовують для перевірки цілісності даних: надсилаючи або зберігаючи файл разом із його хешем, отримувач може обчислити відбиток отриманого файлу й порівняти його з оригіналом — збіг підтверджує, що вміст не зазнав змін. У системах електронного підпису підписують не сам документ, а його хеш; це робить операцію швидшою й економить пропускну здатність, при цьому зберігається гарантія, що будь-яка зміна документа буде помічена.

Для зберігання паролів замість самих паролів у базі зберігають їхні хеші — навіть у разі витоку БД безсольового хешу відновити початковий пароль дуже важко. Щоб зробити атаку ще важчою, до паролів додають «сіль» (salt) і/або застосовують повільні адаптивні хеш-функції або KDF (наприклад, bcrypt, Argon2), що значно ускладнює перебір (brute-force) і атакування через радужні таблиці.

Хешування також використовується для захисту від повторного відтворення повідомлень (replay attacks): одноразові токени і підписи, засновані на хешах і часових мітках або випадкових значеннях, унеможливають багато класів повторних атак. Крім того, відбитки широко застосовують у системах індексації й пошуку (генеруючи компактні ідентифікатори для великих файлів), у дедуплікації даних, у захисті сертифікатів і в блокчейн-технологіях, де хеш-ланцюги гарантують неможливість незаметної зміни транзакцій.

Серед сучасних і надійних алгоритмів хешування — сімейство SHA-2 (наприклад, SHA-256) та SHA-3; вони забезпечують високий рівень стійкості за умови правильної експлуатації. Старі алгоритми, такі як MD5 і SHA-1, визнані ненадійними через практично реалізовані колізії, тому їх слід уникати в захисних сценаріях.

Підсумовуючи, хеш-функції — фундаментальний інструмент інформаційної безпеки: вони дозволяють швидко перевіряти цілісність і автентичність даних, захищати паролі, формувати підставу для цифрових підписів і захищати системи від різних типів атак. Правильний вибір алгоритму, додаткові заходи (salt, KDF) і дотримання сучасних стандартів гарантують, що хешування буде працювати як надійний шар захисту в будь-якій інформаційній системі.

Алгоритми сімейства MD (Message Digest)

Серія хеш-функцій MD, створена Роном Рівестом, призначалася для отримання 128-бітних відбитків повідомлень і довгий час використовувалася для перевірки цілісності та побудови цифрових підписів. Найвідоміший представник — MD5 — історично широко застосовувався, але сьогодні його вразливість до колізій робить його непридатним для криптографічних задач, що вимагають високого рівня безпеки. Сучасні алгоритми повинні забезпечувати складність знаходження двох різних повідомлень з однаковим відбитком, не спиратися на занадто вузькі або ризиковані математичні припущення, бути швидкими в реалізаціях на різних архітектурах і давати високу стійкість до колізій.

MD2

MD2 — старіший алгоритм, розроблений з прицілом на 8-бітні процесори кінця 1980-х. Він виробляє 128-бітний хеш і використовує специфічні контрольні суми та перестановки (частково засновані на цифрах числа π). Через архітектурні особливості MD2 виявився повільним і застарів для сучасних криптографічних потреб; нині його замінюють більш стійкі алгоритми (SHA-2/3).

MD4 та MD5

MD4 заклав базові ідеї для наступних хешів і був орієнтований на швидку обробку на 32-бітних CPU, але згодом виявився вразливим до криптоаналізу.

MD5, як його розвиток, додав ще один раунд обробки та інші зміни, але йому також притаманні слабкі сторони: з 2004 року відомі практичні методи генерації колізій, тому MD5 не рекомендується для захищених застосувань. Сьогодні MD5 іноді використовують лише для некриптографічних контрольних сум.

SHA-1

SHA-1 (Secure Hash Algorithm 1) — наступник SHA-0, спроектований NIST, формує 160-бітний відбиток і довгий час вважався достатньо безпечним. Алгоритм оперує блоками по 512 біт і проходить 80 етапів трансформації з використанням бітових операцій та додавань по модулю. Проте в середині 2000-х були виявлені методи, що значно полегшують пошук колізій, а у 2017 році продемонстровано практичний випадок колізії, через що SHA-1 визнано непридатним для криптографічного використання (підписів, сертифікатів тощо).

SHA-2 і SHA-256

SHA-2 — це сімейство більш сучасних хеш-функцій (включаючи SHA-224, SHA-256, SHA-384, SHA-512 та варіанти SHA-512/224, SHA-512/256), запроваджене NIST як розвиток SHA-1. SHA-256, зокрема, створює 256-бітний відбиток: повідомлення доповнюється до кратного формату, обробляється блоками по 512 біт і проходить 64 внутрішні раунди із застосуванням логічних операцій і додавань по модулю. SHA-2 значно підвищив захищеність у порівнянні з SHA-1 і сьогодні широко використовується в протоколах безпеки та блокчейнах. Деякі варіанти SHA-2 (наприклад, SHA-512) вимагають більше обчислювальних ресурсів через довші слова й більші внутрішні стани.

SHA-3

SHA-3 — нове покоління стандарту хешування від NIST, побудоване на іншому принципі (конструкція sponge, «губка»), що відрізняє його архітектуру від SHA-2. SHA-3 пропонує варіанти з різними довжинами виходу (SHA3-224/256/384/512) та адаптивні розширення SHAKE (SHAKE128/256) з довільною довжиною виводу. Конструкція «губка» дає гнучкість і протилежні риси до атак, від яких потерпали попередники; водночас SHA-3 іноді є менш ефективним у деяких реалізаціях порівняно з SHA-2, тому його впровадження поступове, але воно надає додатковий шар захисту й універсальні можливості застосування.

RIPEMD

RIPEMD (Race Integrity Primitives Evaluation Message Digest) — серія хешів, розроблена в Європі як альтернатива MD і SHA-1. Існують версії RIPEMD-128 та RIPEMD-160 (а також пізніші варіанти RIPEMD-256/320). RIPEMD-160 довгий час вважався більш безпечним, ніж MD5 і SHA-1, і використовувався в деяких системах і криптовалютах. Однак із поширенням

SHA-2/3 його застосування скорочується, і для нових проєктів зазвичай радять віддавати перевагу сучаснішим стандартам.

Підсумок і практичні рекомендації

- Алгоритми сімейства MD (MD2/MD4/MD5) нині вважаються застарілими для криптографічного захисту через проблеми зі стійкістю до колізій.
- SHA-1 також випав із переліку безпечних хешів для підписів і сертифікацій.
- SHA-2 (особливо SHA-256) — надійна і широко підтримувана опція для більшості застосувань; SHA-3 пропонує альтернативну архітектуру та додаткову гнучкість.
- RIPEMD-160 може використовуватися для сумісності в спадкових системах, але для нових розробок краще обирати SHA-2/3.

У практиці слід обирати сучасні стандартні алгоритми (SHA-256/512 або SHA-3), уникати MD5 і SHA-1 у криптографічних сценаріях, застосовувати додаткові заходи (salt, KDF, повільні хеші для паролів), і слідкувати за рекомендаціями стандартів та актуальними дослідженнями з криптоаналізу.

Електронний цифровий підпис (ЕЦП) — це механізм, що дозволяє підтвердити автентичність автора документа, гарантувати цілісність вмісту і запобігти відмові від підпису; реалізується він за допомогою асиметричної криптографії, коли кожний користувач має приватний ключ (що залишається в секреті для створення підписів) і публічний ключ (яким користуються інші для перевірки). Загальна процедура проста: повідомлення спочатку стискають або хешують, потім отриманий хеш підписують приватним ключем відправника; отримувач самостійно обчислює хеш повідомлення й, застосувавши публічний ключ, перевіряє відповідність — у разі збігу підтверджується і цілісність, і походження повідомлення.

Серед найпоширеніших реалізацій ЕЦП — RSA, DSA, ECDSA, EdDSA і ElGamal, кожен з яких має свої математичні основи й практичні властивості. RSA базується на модульній арифметиці й складності факторизації великих чисел; у підписній схемі зазвичай хеш повідомлення трансформується приватним експонентом і перевіряється публічною експонентою, що дає високу сумарну стійкість, але робить операції відносно ресурсомісткими при великих ключах. DSA опирається на проблему дискретного логарифма і забезпечує інший формат підпису з перевагою швидшого формування підписів у деяких реалізаціях, проте має обмеження щодо структури підписів і не призначений для шифрування даних.

Більш ефективні в просторі та швидкодії алгоритми будуються на еліптичних кривих: ECDSA реалізує ідеї DSA у групі точок еліптичної кривої, забезпечуючи той самий рівень захисту при значно менших ключах і підписах

— це робить її привабливою для мобільних і вбудованих платформ та високонавантажених сервісів. EdDSA — сучасніший підхід на основі кривих Едвардса, проєктований для безпечної й ефективної реалізації: він використовує детерміністичний процес формування випадкових значень, що знижує ризики, пов'язані з неналежною генерацією випадковості, і загалом забезпечує вищу опірність до атак побічних каналів. ElGamal теж базується на дискретному логарифмі, має сильні теоретичні властивості, але практично часто поступається іншим схемам через більший розмір підписів і складність впровадження.

При виборі конкретного алгоритму слід орієнтуватися на практичні вимоги: RSA дає широку сумісність, ECC-варіанти — кращу продуктивність і економію трафіку/пам'яті, EdDSA поєднує швидкість із захищеністю реалізації, а ElGamal підходить для специфічних сценаріїв. Незалежно від алгоритму, надійність системи підписів залежить не лише від математичної основи, а й від інженерних аспектів: генерація ключів повинна відбуватися з достатньою ентропією, приватні ключі треба зберігати у захищених сховищах (HSM/KMS), а операції підписування і верифікації — виконувати в constant-time, щоб мінімізувати ризики побічних каналів. Важливим також є правильне застосування хеш-функцій і схем паддінгу, аби уникнути відомих векторів атак.

Універсальні стандарти і нормативи забезпечують сумісність і мінімальні вимоги безпеки: до них належать FIPS (зокрема FIPS PUB 186 для DSA/ECDSA/RSA), ISO/IEC 14888 для схем цифрових підписів, X.509 для формування та управління сертифікатами у PKI, а також численні RFC, що визначають використання підписів у інтернет-протоколах. Дотримання цих стандартів, використання перевірених бібліотек і сертифікованих апаратних модулів є обов'язковою практикою для забезпечення реальної безпеки й інтероперабельності.

Переваги ЕЦП очевидні: вони забезпечують цілісність даних, автентичність автора і неможливість відмови від підпису. При цьому існують і обмеження — висока обчислювальна вартість для великих повідомлень (особливо у RSA), чутливість до довжини та якості ключів, а також ризики через неправильну реалізацію (наприклад, слабка генерація випадкових чисел у DSA/ECDSA може призвести до компрометації приватного ключа). Крім того, класичні асиметричні схеми під загрозою появи універсальних квантових комп'ютерів, тому сучасні системи мають проєктуватися з урахуванням криптоагільності — можливості перейти на постквантові або гібридні підходи в майбутньому.

Отже, електронні цифрові підписи залишаються фундаментальною технологією для юридично значущої фіксації та безпечної передачі електронних документів; вибір конкретної схеми та її безпечне впровадження

повинні базуватися на вимогах до продуктивності, розмірів підписів, сумісності зі стандартами і довгостроковій стійкості проти нових загроз.

Цифрові підписи і контроль цілісності інформації — це взаємопов'язані механізми, що разом забезпечують довіру до електронних даних: перший гарантує автентичність і неможливість заперечення автора повідомлення, другий підтверджує, що вміст не був змінений під час зберігання або передачі. У практичній реалізації обидва підходи опираються на криптографічні примітиви — хеш-функції для створення компактного «відбитка» даних і асиметричні алгоритми для підписування цього відбитка приватним ключем. Така комбінація дозволяє підписувати великі документи ефективно: замість підпису всього файлу обчислюється його хеш, підписується хеш, а при перевірці отримувач обчислює хеш отриманого документу і порівнює його з розкритим підписом.

Контроль цілісності починається з криптографічного хешування: надійна хеш-функція (наприклад, SHA-256 або SHA-3) генерує фіксовану довжину відбитка, який змінюється радикально навіть при найменшій модифікації вхідних даних. Для внутрішніх перевірок і високочастотних операцій іноді застосовують також Message Authentication Codes (HMAC), які поєднують хеш з секретним ключем і забезпечують одночасно цілісність і автентифікацію у випадках symmetric trust. Однак у контексті юридично значимих підписів і міжорганізаційних транзакцій перевага надається асиметричним підходам — цифровим підписам, бо вони дають можливість верифікації будь-якому сторонньому учаснику без потреби в обміні секретами.

Архітектурно процес забезпечення цілісності й підпису виглядає так: спочатку документ або повідомлення проходить етап хешування; отриманий хеш підписується приватним ключем підписувача (за допомогою RSA-PSS, ECDSA, EdDSA чи іншої сертифікованої схеми); разом із самим документом і підписом може додаватися сертифікат підписувача, що пов'язує публічний ключ з ідентичністю (PKI, X.509). При перевірці відправлене тіло документу хешується заново, обчислений хеш порівнюється з тим, який відновлено з підпису за допомогою публічного ключа; збіг значень підтверджує як цілісність, так і походження повідомлення.

Для підвищення надійності в реальних системах застосовують додаткові механізми: часові печатки (timestamping) для фіксації моменту підписування, що важливо для юридичної сили й для політик зберігання; ланцюги сертифікації і CRL/OCSP для перевірки статусу сертифікатів; використання HSM/KMS для зберігання і виконання приватних ключів, щоб запобігти їх витоку; і мультипідписні або threshold-схеми для розподілу довіри при критичних операціях. Також поширені практики підписування метаданих та побічних

артефактів (наприклад, контрольні суми пакетів, журнали транзакцій), щоб гарантувати цілісність не лише основного документа, а й супутніх даних.

Юридичний та процедурний аспекти не менш важливі: цифрові підписи мають відповідати нормативним вимогам (електронна ідентифікація, вимоги до РКІ, зберігання доказів), політикам аудитів і збереженню доказів у випадку спорів (наприклад, збереження логів доступу, тимчасових печаток і метаданих підписів). Для систем, де зберігається інформація довготривало (архіви, медичні записи), треба планувати «cryptographic agility» — можливість змінити алгоритми і ротацію ключів, а також передбачати захист від загроз «collect-now-decrypt-later», зокрема через перехід на постквантові або гібридні схеми, якщо потрібна конфіденційність у десятиліттях.

Слабкі місця реалізації частіше пояснюються не математикою, а інженерією: неправильно підібрані або застарілі хеш-алгоритми (MD5, SHA-1), відсутність salt та KDF для паролів, вразливі реалізації (не constant-time), ненадійна генерація випадкових чисел для підпису (що може зламати DSA/ECDSA) або ненадійне зберігання приватних ключів. Тому практичні рекомендації включають: використання сучасних стандартів (RSA-PSS / ECDSA / EdDSA з рекомендованими параметрами), застосування SHA-2/3 для хешування, зберігання ключів у HSM/KMS, ретельну політику ротації та відкликання ключів, логування і аудит операцій підписування та верифікації.

У підсумку, цифрові підписи у поєднанні з механізмами контролю цілісності формують надійний каркас довіри в електронному обміні: вони дозволяють гарантувати, що дані не були змінені, що їх підписав певний суб'єкт, і що підпис не може бути відкинутий. Забезпечення цих властивостей вимагає як правильного вибору криптографічних примітивів і стандартів, так і дисциплінованої інженерної практики — без цього навіть математично надійні алгоритми не забезпечать реальної безпеки.

5.5. ПРОТОКОЛИ БЕЗПЕЧНОГО ОБМІНУ ДАНИМИ (SSL/TLS 1.3, QUIC, IPSEC, WIREGUARD)

Протоколи безпечного обміну даними — це набір стандартів і практик, які забезпечують конфіденційність, цілісність і автентичність мережеских з'єднань; серед найважливіших сьогодні — TLS 1.3 (історично «SSL/TLS»), QUIC, IPsec і WireGuard, кожен із яких має свої архітектурні рішення, призначення й сильні сторони. TLS 1.3 — сучасна версія протоколу для захисту потоків TCP-трафіку (HTTPS тощо) — спрощує та пришвидшує handshake у порівнянні зі старими версіями: він обов'язково вимагає використання ефемерних ключів (ECDHE чи інші DH-варіанти) для forward secrecy, переважно оперує AEAD-шифрами (наприклад AES-GCM, ChaCha20-Poly1305) і значно скоротив кількість кроків

для встановлення сесії, що знижує затримку при встановленні захищеного з'єднання. TLS 1.3 також видалив багато застарілих і вразливих можливостей (старі шифронабори, слабкі хеші, RSA-key-exchange), спрощуючи конфігурацію та підвищуючи безпеку загалом. Упровадження TLS потребує PKI/сертифікатів (X.509) для автентифікації серверів (і опційно клієнтів), правильного управління сертифікатами, вибору сучасних шифрів і налаштування параметрів (наприклад, OCSP/CRL для перевірки відкликання, коректних таймаутів і політик щодо повторного використання сесій або 0-RTT механізмів з урахуванням ризиків відтворення).

SSL / TLS — це протоколи, які забезпечують захищене з'єднання між клієнтськими додатками та серверами в мережі, гарантують конфіденційність переданих даних, їхню цілісність і автентичність учасників зв'язку. Історично першим був SSL, але згодом він був замінений і розвинений у вигляді сімейства протоколів TLS; сучасні впровадження використовують саме TLS, оскільки він усуває слабкі місця попередніх версій і пропонує поліпшену криптографічну стійкість.

Механізм роботи базується на поєднанні асиметричної криптографії для встановлення довірчого контексту й симетричного шифрування для ефективною передачі даних. На початку сесії відбувається handshake — обмін повідомленнями, під час якого сторони узгоджують набори шифрів, перевіряють сертифікати й виробляють спільний секрет (сесійний ключ). Після завершення цього етапу весь подальший трафік шифрується симетричними AEAD-алгоритмами (що одночасно забезпечують і конфіденційність, і цілісність), а також супроводжується криптографічними кодами цілісності і механізмами автентифікації повідомлень.

Ключовим елементом інфраструктури TLS є модель PKI: цифрові сертифікати, видані довіреними центрами сертифікації (CA), зв'язують публічні ключі з ідентичностями серверів (і за потреби — клієнтів). Перевірка підпису сертифіката дозволяє клієнту упевнитися, що він спілкується саме з тим сервером, за допомогою якого пов'язаний публічний ключ, і тим самим мінімізується ризик MITM-атак. Сертифікати містять метадані (інформацію про власника, період дії тощо) і можуть бути відкликані або перевірені через OCSP/CRL механізми.

Версії протоколів еволюціонували: від ранніх SSL 2.0/3.0 із відомими вразливостями до декількох ітерацій TLS, де TLS 1.3 є нині рекомендованою та найбільш безпечною специфікацією. TLS 1.3 скоротив і спростив handshake, вимагав використання ефемерних ключів для забезпечення forward secrecy, прибрав слабкі або застарілі криптосхеми та за замовчуванням орієнтується на AEAD-режими. Через це підключення встановлюються швидше та безпечніше, ніж у попередніх версіях.

Архітектурно протокол складається з двох головних шарів: handshake (встановлення параметрів і аутентифікація) та record layer (пакетизація, шифрування і перевірка цілісності самих повідомлень). Під час handshake відбуваються вибір шифропакета, обмін сертифікатами й встановлення сесійних ключів; далі record layer використовує ці ключі для захисту кожного повідомлення.

Цифрові сертифікати можуть видаватися централізовано СА, бути самопідписаними або застосовуватися у внутрішніх (тестових) сценаріях; у виробничому середовищі зазвичай вимагають довірених СА та політик управління сертифікатами — їх отримання, ротація й відкликання є частиною операційної безпеки. Для підвищення надійності також застосовують часові печатки, механізми перевірки статусу сертифікатів і зберігання приватних ключів у захищених модулях (HSM/KMS).

TLS/SSL використовуються не лише для захисту веб-трафіку через HTTPS, але й для безпечної роботи протоколів електронної пошти (SMTP/IMAP), FTP, VPN і багатьох інших сервісів. Проте важливо пам'ятати: безпечність залежить не тільки від специфікації протоколу, а й від правильної конфігурації — відключення старих шифрів, застосування рекомендованих параметрів, обробки 0-RTT та інших функцій з урахуванням ризиків, і постійного моніторингу й оновлення стеку криптографії.

Отже, TLS (як спадкоємець SSL) є базовим механізмом сучасного захисту мережових з'єднань: він поєднує симетричну й асиметричну криптографію, цифрові сертифікати й стандарти управління ключами для гарантування конфіденційності, цілісності та автентичності комунікацій у мережі.

QUIC — це транспортний протокол, що поєднує механізм встановлення сесії та шифрування (побудований над UDP) і у багатьох реалізаціях інтегрує TLS 1.3 як механізм шифрування/аутентифікації handshake. Основна мета QUIC — зменшити латентність та уникнути проблем TCP (head-of-line blocking) шляхом мультиплексування кількох потоків на одній сесії і швидкого відновлення після зміни мережових умов (наприклад, міграція IP при перемиканні між Wi-Fi і мобільною мережею). Завдяки використанню UDP-основаного підходу QUIC дає можливість здійснювати 0-RTT або 1-RTT handshakes для вже відомих клієнтом серверів, що пришвидшує відкриття з'єднань; при цьому гарантує ті самі криптографічні властивості TLS 1.3 (AEAD, forward secrecy). QUIC краще підходить для сучасних веб-додатків, стрімінгу та реального часу, але його використання може зустрічати обмеження в мережах із суворими UDP-фільтрами або старими середовищами, де мережеві пристрої очікують TCP-семантики.

IPsec — набір протоколів для захисту мережевого трафіку на рівні мережевого стеку (IP). IPsec може працювати в двох режимах: транспортному

(шифрується тільки корисне навантаження пакету) і тунельному (шифрується весь IP-пакет і обертається новим заголовком) — що робить його основним інструментом для побудови VPN і захисту міжмережових шлюзів. IPsec складається з механізмів для встановлення захисних асоціацій (IKEv2 як протокол контролю і обміну ключами), а також протоколів захисту даних (ESP — Encapsulating Security Payload для шифрування й автентифікації; AH — Authentication Header — рідше використовується). Перевага IPsec — гнучкість (можливість політичного фільтрування, тунелювання цілих підмереж), підтримка на рівні ОС і мережевого обладнання; недолік — складність конфігурації, потенційні проблеми з NAT (хоча сучасні розширення і NAT-Traversal/UDP-encapsulation вирішують багато проблем), і іноді більш висока затримка при великих накладних витратах на тунелювання.

IPsec — це набір відкритих стандартів для захисту IP-трафіку, що забезпечує конфіденційність, автентичність і цілісність пакетів при їх передачі по інтернету або в приватних IP-мережах. Технологія дозволяє встановлювати захищені канали між кінцевими вузлами або між мережевими шлюзами за допомогою криптографічних механізмів: шифрування для приховування вмісту, підписів або MAC для перевірки походження і цілісності, а також засобів управління ключами для безпечної організації сеансів зв'язку.

Ядро IPsec складається з кількох логічних компонентів. Два основні протоколи — AH (Authentication Header) і ESP (Encapsulating Security Payload) — реалізують захисні функції на рівні IP-пакета: AH додає криптографічний заголовок, що підтверджує автентичність і незмінність певних полів пакета (проте не шифрує корисне навантаження), а ESP може забезпечувати шифрування, а також автентифікацію й перевірку цілісності даних, тобто надає повніший набір засобів для конфіденційного обміну. Підтримується два режиму роботи: транспортний режим, коли захищається тільки корисне навантаження, і тунельний режим, у якому вміст усього початкового IP-пакета інкапсулюється й захищається новим заголовком — цей режим зазвичай застосовують для VPN-тунелів між мережами.

Організацію й узгодження параметрів захисту в IPsec забезпечує механізм встановлення безпечних асоціацій (Security Associations, SA). SA визначає, які алгоритми і ключі використовуються для шифрування та аутентифікації конкретного потоку трафіку; оскільки SA є односпрямованими, для двостороннього захищеного зв'язку створюються дві асоціації — по одній у кожному напрямку. Процес установа SA зазвичай автоматизується протоколом IKE (Internet Key Exchange), який в рамках переговорів обмінюється параметрами, проводить аутентифікацію сторін (сертифікати, попередньо спільні ключі тощо) і домовляється про ключі сеансу.

IPSec реалізований таким чином, щоб бути гнучким у виборі криптопримітивів: сучасні інсталяції використовують швидкі й безпечні режими AEAD (наприклад AES-GCM) або інші рекомендовані алгоритми шифрування й хешування, замінивши застарілі варіанти. Ця гнучкість дозволяє адаптувати захист до регуляторних обмежень, продуктивнісних вимог і потреб сумісності з різними ОС та мережевим обладнанням.

Архітектурно кожен IPSec-вузол оперує двома важливими таблицями: базою політик безпеки (SPD, Security Policy Database), яка задає правила застосування захисту до вхідних і вихідних пакетів (на основі селекторів: адреси, порти, протокол тощо), та базою безпечних асоціацій (SAD, Security Association Database), що зберігає актуальні параметри SA — SPI, ключі, лічильники пакетів і інші атрибути. Коли пакет відповідає правилу в SPD, він обробляється відповідною SA з SAD; якщо потрібної SA ще немає, її можна створити динамічно через IKE.

IPSec застосовується у різних архітектурах: хост-хост (прямий захист між кінцевими пристроями), шлюз-шлюз (тунелі між мережами) та хост-шлюз (віддалений користувач підключається до мережі через шлюз). Така універсальність робить IPSec стандартним рішенням для корпоративних VPN, захисту міжмережових з'єднань і захищеного доступу до хмарних ресурсів. При цьому вибір режиму (транспортний vs тунельний) і схеми розгортання визначається політиками безпеки й вимогами до маршрутизації та продуктивності.

Слід зауважити, що реалізація IPSec вимагає правильної організації керування ключами, політик відкликання та ротації, захищеного зберігання приватних ключів (HSM/KMS) і постійного оновлення криптонаборів у відповідь на розвиток криптоаналізу. При належному конфігуруванні та використанні сучасних алгоритмів IPSec зберігає високу релевантність для захисту IP-мереж і продовжує широко використовуватися у виробничих середовищах як на IPv4, так і на IPv6 стеку.

WireGuard — відносно новий легкий VPN-протокол, спроектований для простоти, продуктивності та безпечності за замовчуванням. Він використовує сучасні криптопримітиви (Curve25519 для ECDH, ChaCha20-Poly1305 або XChaCha20/Poly1305 для шифрування та Poly1305 для автентифікації, Blake2s для хешування) і мінімальний кодовий базис, що спрощує аудит і зменшує поверхню атак. WireGuard працює поверх UDP і управляє відносинами «peer-to-peer», де кожен реєр має статичну пару ключів; тунелювання устроїв відбувається через прості правила маршрутизації, і WireGuard добре працює в умовах мобільних мереж (швидка реконфігурація, низька затримка). Головні плюси WireGuard — простота налаштування, висока продуктивність та

сучасний криптонабір; мінус — спочатку не було складної інфраструктури PKI (хоча керування ключами та інтеграція з системами аутентифікації вирішуються зовнішніми інструментами), і збереження статичних ключів на клієнтах потребує управляючих політик (HSM/KMS/скрипти ротації) у великих інсталяціях.

Усі ці протоколи мають спільні та відмінні риси. TLS 1.3 орієнтований на захист на рівні додатків (HTTPS, SMTP, інші TLS-застосунки) і інтегрується в практично всі сучасні веб-стеки; QUIC переносить TLS 1.3 на транспортний рівень поверх UDP і додає оптимізації для веб-latency та мультиплексування; IPsec — стандарт для мережевого рівня з широкими можливостями тунелювання і політичного контролю; WireGuard пропонує просте, швидке та безпечне VPN-рішення з мінімальною складністю. При виборі між ними слід керуватися цілями: для захищених веб-застосунків і API — TLS 1.3 або QUIC (HTTP/3); для побудови корпоративних VPN-тунелів між мережами з політичними правилами — IPsec; для легких, продуктивних VPN-з'єднань користувач-до-мережі або вузол-до-вузла — WireGuard.

У виробничому середовищі важливо: використовувати лише сучасні шифронабори й AEAD-режими, відключати застарілі опції, забезпечити коректне управління сертифікатами й ключами (HSM/KMS, автоматизація ротації), налаштувати моніторинг і логування TLS/QUIC/ IPsec подій, та тестувати поведінку під навантаженням і в умовах непередбачуваних мережевих змін. Додаткові міркування включають підтримку NAT-Traversal для VPN, політики DPI/ACL на мережевих пристроях, та сумісність з пристроями проміжної інфраструктури (load balancer, WAF, CDN). Питання 0-RTT (швидкий відновлювальний трафік) в TLS/QUIC дає переваги по затримці, але вимагає уваги до ризиків повторного відтворення (replay) і до політик автентифікації.

У контексті квантової загрози потрібно планувати криптоагільні підходи: проектувати можливість оновлення KEX/шифрів у TLS/QUIC, підтримувати гібридні handshake (класичний + PQC) там, де потрібна довготривала конфіденційність, і оцінювати вплив збільшених розмірів ключів/підписів на продуктивність і мережевий трафік. Також варто враховувати налаштування захисту від атак на рівні реалізацій (побічні канали, неправильні параметри криптографічних бібліотек) та продовжувати оновлювати конфігурації згідно з рекомендаціями IETF/NIST.

TLS 1.3, QUIC, IPsec і WireGuard — це інструменти для різних задач захисту мережі: TLS 1.3/QUIC — для захисту потоків додатків і веб-трафіку з акцентом на мінімальну латентність і сучасні криптографічні практики; IPsec — для гнучкого захисту мережевого рівня і побудови корпоративних тунелів; WireGuard — для простих, швидких і безпечних VPN-включень. Вибір і

безпечне впровадження вимагають уваги до криптопараметрів, управління ключами, сумісності з інфраструктурою та планування щодо майбутніх квантових загроз.

5.6. СУЧАСНІ НАПРЯМИ РОЗВИТКУ КРИПТОГРАФІЇ: КВАНТОВА ТА ГОМОМОРФНА КРИПТОГРАФІЯ)

Сучасні напрями розвитку криптографії охоплюють дві великі, але різні гілки — квантову криптографію (у двох значеннях: C1 — криптографія, що використовує квантову фізику, і C2 — класична «постквантова» криптографія, стійка до квантових комп'ютерів) та гомоморфну криптографію (здатність оперувати над зашифрованими даними). Разом ці підходи формують напрямки досліджень і інженерії, що вирішують фундаментальні завдання: як зберегти конфіденційність і цілісність у епоху квантових обчислень і як дозволити корисні обчислення над зашифрованими наборами даних без розшифровки.

Почнемо з квантової тематики. По-перше, під «квантовою криптографією» інколи розуміють **криптографію на основі квантової фізики**, найпопулярнішим прикладом якої є QKD (Quantum Key Distribution, напр., протокол BB84). QKD використовує властивості квантових станів (неможливість копіювання, колапс стану при вимірюванні) для встановлення спільного секретного ключа з інформацією про можливі підслуховування у вигляді підвищення рівня помилок; він дає інформаційну (теоретичну) гарантію секретності каналу ключів, але вимагає спеціального квантового обладнання, каналів з малою загубою і має інженерні обмеження щодо відстані й пропускну здатності. По-друге, іще важливіше в практиці — **постквантова (quantum-resistant) криптографія**, тобто класичні алгоритми, розроблені так, щоб їхня безпека не руйнувалась алгоритмами квантових комп'ютерів (наприклад, Шором). Сюди належать сімейства на основі решіток (lattice-based), кодування (code-based), мультिवаріантних рівнянь, хеш-орієнтовані підписи та інші; ці підходи активно стандартизуються і впроваджуються (NIST-процес постквантової стандартизації — приклад). Головна ідея — замінити примітиви, вразливі до квантових атак, на такі, що опираються на завдання, для яких немає відомих ефективних квантових алгоритмів.

Гомоморфна криптографія — це концепція і практичні схеми, що дозволяють виконувати обчислення над шифртекстами так, щоб результат після розшифрування співпадав з результатом обчислення над відкритим текстом. Є три класи: **частково гомоморфні** схеми (підтримують одну операцію — додавання або множення; приклади: Paillier — адитивна, RSA/ElGamal у певних режимах — множні), **рівневі/leveled FHE** (дозволяють виконувати обчислення

до певної глибини без bootstrapping), і повністю гомоморфні шифри (FHE, **Fully Homomorphic Encryption**), що підтримують довільні комбінації операцій завдяки техніці bootstrapping. З моменту першого практично доказаного FHE-конструкта Гентрі (2009) з'явилося кілька сімейств реалізацій: BGV, BFV, CKKS тощо; CKKS важливий для приблизних чисельних обчислень (машинне навчання над зашифрованими моделями/даними). Практичне застосування гомоморфного шифрування дозволяє побудувати сервіси «шифрованих обчислень» — наприклад, приватні аналітики, аутсорсинг обробки даних у хмарі без доступу до відкритих даних, приватні ML-інференси — але ці системи дорогої в обчислювальному та пам'ятному вимірі й вимагають ретельної оптимізації (батчинг, параметризація, скорочення точності там, де це прийнятно).

Обидва напрями мають сильні точки і обмеження, і в реальних системах часто застосовують гібридний підхід. Наприклад, постквантові алгоритми вже використовують у поєднанні з класичними схемами для плавної міграції; QKD — там, де потрібні інформаційно-теоретичні гарантії і є можливість розгорнути квантову інфраструктуру (фінансові центри, державні канали); а гомоморфні методи — для специфічних випадків приватних обчислень, де латентність і вартість виправдані бізнес-перевагами. Також гомоморфія чудово комбінується з іншими приватними протоколами — MPC (secure multi-party computation) і TEEs (trusted execution environments) — щоб знайти компроміс між ефективністю й рівнем довіри.

Технічні проблеми й напрямки досліджень сьогодні фокусуються на підвищенні ефективності та практичної придатності: для постквантових схем — зниження розмірів ключів/підписів, покращення швидкодії та підвищення стійкості параметрів; для QKD — подолання обмежень дальності й інтеграція з класичними мережами (trusted nodes, квантові репітери); для гомоморфії — швидший bootstrapping, оптимальні параметри шуму, апаратне прискорення (GPU/FPGA/ASIC), стандартизація API і бібліотек, а також розробка алгоритмів, толерантних до обмеженої точності (CKKS). Етичні й регуляторні аспекти також важливі: постквантова міграція має плануватись заздалегідь (щоб уникнути ризику «collect-now-decrypt-later»), а приватні обчислення повинні враховувати законодавство щодо обробки персональних даних.

Підсумовуючи, «квантова криптографія» сьогодні означає два взаємопов'язані напрями: технології, що використовують квантову фізику для захисту ключів (QKD), і розробку класичних криптосхем, стійких до квантових атак (PQC). Гомоморфна криптографія відкриває нові можливості для безпечної обробки даних без їх розкриття, але потребує подальшого підвищення ефективності для масового прийняття. Разом ці напрями формують майбутнє захисту інформації: один вирішує загрози нового обчислювального ландшафту,

інший — дозволяє нові моделі приватності і сервісів над даними. Для інженерів і архітекторів важливими практичними кроками залишаються планування постквантової міграції, валідація бібліотек/реалізацій, та експериментальне впровадження гомоморфних прототипів у прикладних задачах, де їхня користь переважає витрати.

СПИСОК ДЖЕРЕЛ РОЗДІЛУ 5

1. Костюк Ю.В., Складанний П.М., Бебешко Б.Т., Хорольська К.В., Рзаєва С.Л., Ворохоб М.В. (2025) Безпека інформаційно-комунікаційних систем. Київський університет імені Бориса Грінченка, Україна, Київ. - 1017 с.
2. Костюк Ю.В., Складанний П.М., Гулак Г.М., Бебешко Б.Т., Хорольська К.В., Рзаєва С.Л. (2025) Системи захисту інформації. Київський університет імені Бориса Грінченка, Україна, Київ. - 886с.
3. Kostiuk, Yu. V., Skladannyi, P. M., Bebeshko, B. T., Khorolska, K. V., Rzaieva, S. L., & Vorokhob, M. V. (2025). Information and communication systems security. [Textbook] Kyiv: Borys Grinchenko Kyiv Metropolitan University.
4. Kostiuk, Yu. V., Skladannyi, P. M., Hulak, H. M., Bebeshko, B. T., Khorolska, K. V., & Rzaieva, S. L. (2025). Information security systems. [Textbook] Kyiv: Borys Grinchenko Kyiv Metropolitan University.
5. Гулак Г. М., Жильцов О. Б., Киричок Р. В., Коршун Н. В., Складанний П. М. Інформаційна та кібернетична безпека підприємства : підруч. / Г. М. Гулак, О. Б. Жильцов, Р. В. Киричок, Н. В. Коршун, П. М. Складанний – Львів : Видавець Марченко Т. В., 2023. – 370 с.
6. Zhurakovskiy, B. and Otrokh, S. and Poliakov, M. and Poliakov, O. and Skladannyi, Pavlo (2024) *Enhancing information transmission security with stochastic codes* Classic, Quantum, and Post-Quantum Cryptography 2024, 3829. pp. 62-69. ISSN 1613-0073
7. Kriuchkova, Larysa and Tsmokanych, Ivan and Shevchenko, Svitlana and Bohdanov, Oleksandr and Mazur, Nataliia (2023) *Experimental Research of the Parameters of Danger and Protective Signals Attached to High-Frequency Imposition* Cybersecurity Providing in Information and Telecommunication Systems, 3550. pp. 261-268. ISSN 1613-0073
8. Berestov, Denis and Kurchenko, Oleg and Zubyk, L. and Kulibaba, Serhii and Mazur, Nataliia (2023) *Assessment of Weather Risks for Agriculture using Big Data and Industrial Internet of Things Technologies* Cybersecurity

Providing in Information and Telecommunication Systems II 2023, 3550. pp. 1-13. ISSN 1613-0073

9. Tkachenko, K. and Tkachenko, O. and Tkachenko, O. and Mazur, Nataliia and Mashkina, Iryna (2024) *Ontological Approach in Modern Educational Processes* Cybersecurity Providing in Information and Telecommunication Systems 2024, 3654. pp. 88-97. ISSN 1613-0073

10. Popereshnyak, S. and Novikov, Y. and Zhdanova, Yuliia (2024) *Cryptographic system security approaches by monitoring the random numbers generation* Cybersecurity Providing in Information and Telecommunication Systems II 2024, 3826. pp. 301-309. ISSN 1613-0073

11. Mykhaylova, O. and Stefankiv, A. and Nakonechny, T. and Fedynyshyn, T. and Sokolov, Volodymyr (2024) *Resistance to Replay Attacks of Remote Control Protocols using the 433 MHz Radio Channel* Cybersecurity Providing in Information and Telecommunication Systems 2024, 3654. pp. 98-110. ISSN 1613-0073

12. Novytskyi, Anton and Sokolov, Volodymyr and Kriuchkova, Larysa and Skladannyi, Pavlo (2025) *Determining the error distribution of BLE beacons at antenna near and far fields* Cyber Hygiene & Conflict Management in Global Information Networks 2025, 4024. pp. 133-143. ISSN 1613-0073

13. Romanovskiy, O. et al. (2025). Accuracy Improvement of Spoken Language Identification System for Close-Related Languages. In: Hu, Z., Yanovsky, F., Dychka, I., He, M. (eds) *Advances in Computer Science for Engineering and Education VII. ICCSEEA 2024* 2024. Lecture Notes on Data Engineering and Communications Technologies, vol 242. Springer, Cham. https://doi.org/10.1007/978-3-031-84228-3_4

14. Kriuchkova, Larysa and Tsmokanych, I. and Vovk, M. and Mazur, Nataliia and Bohdanov, Oleksandr (2024) *Influence of Protective Signals on Dangerous Signals of High-Frequency Imposition* Cybersecurity Providing in Information and Telecommunication Systems 2024, 3654. pp. 419-425. ISSN 1613-0073

15. Buhas, V. and Ponomarenko, I. and Buhas, N. and Hulak, Hennadii (2024) *Cybersecurity Role in AI-Powered Digital Marketing* Digital Economy Concepts and Technologies Workshop 2024, 3665. pp. 1-11. ISSN 1613-0073

16. Dreis, Y. and Korchenko, O. and Sokolov, Volodymyr and Skladannyi, Pavlo (2024) *Model to formation data base of secondary parameters for assessing status of the state secret protection* Cyber Security and Data Protection 2024 (3800). pp. 1-11. ISSN 1613-0073

17. Shevchenko, Svitlana & Zhdanova, Yuliia & Astapenya, Volodymyr & Nehodenko, Olena & Spasiteleva, Svitlana. (2024). *Conflicting Subsystems in the Information Space: A Study at the Software and Hardware Levels*.
18. Kolbasova, K and Zhurakovskiy, Bohdan and Poltorak, V and Nakonechnyi, V and Kyrychok, Roman (2023) *Smart Home Network based on Cisco Equipment* Cybersecurity Providing in Information and Telecommunication Systems, 3550. pp. 70-80. ISSN 1613-0073
19. Virovets, Denis and Obushnyi, Sergiy and Ramskyi, Andrii and Hulak, Hennadii (2024) *Soul Bound Token as Digital Twins in Peer-to-Peer Economic Systems* Digital Economy Concepts and Technologies Workshop 2024, 3665. pp. 97-105. ISSN 1613-0073
20. Tyshyk, I. and Hulak, Hennadii (2024) *Testing an organization's information system for unauthorized access* Cybersecurity Providing in Information and Telecommunication Systems II 2024, 3826. pp. 17-29. ISSN 1613-0073
21. Zhebka, Serhii and Zhebka, Viktoriia and Hulak, Hennadii and Kyrychok, Roman and Platonenko, Artem (2025) *Method for Adaptive Allocation of Cryptographic Resources in Distributed Databases* Cybersecurity Providing in Information and Telecommunication Systems 2025 (3991). pp. 620-628. ISSN 1613-0073
22. Dovzhenko, Nadiia & Ivanichenko, Yevhen & Skladannyi, Pavlo & Zhylytsov, Oleksii. (2024). *Research of UAV and sensor network integration features for routing optimization and energy consumption reduction*.
23. Zhebka, V. and Skladannyi, Pavlo and Bazak, Y and Bondarchuk, A. and Storchak, K. (2024) *Methods for Predicting Failures in a Smart Home* Digital Economy Concepts and Technologies Workshop 2024, 3665. pp. 70-78. ISSN 1613-0073
24. Kriuchkova, Larysa and Tsmokanych, I. and Mazur, Nataliia and Tarasenko, D. and Osadcha, V. (2024) *Experimental determination of protective signal parameters for effective "swinging" of the carrier frequency of high-frequency imposition* Cybersecurity Providing in Information and Telecommunication Systems II 2024, 3826. pp. 251-259. ISSN 1613-0073
25. Kraus, N. and Kraus, K. and Shevchenko, Svitlana and Proshkin, Volodymyr (2024) *Digitalizing Production and Industry through 4.0 Technologies Implementation* Digital Economy Concepts and Technologies Workshop 2024, 3665. pp. 36-47. ISSN 1613-0073
26. Kulibaba, S. and Zubyk, L. and Kurchenko, O. and Sinitsyn, I. and Mazur, Nataliia (2023) *Linked System of Data Organization and Management* Cybersecurity Providing in Information and Telecommunication Systems (3421). pp. 117-126. ISSN 1613-0073

27. Chyzhevskaya, Maryna & Романовська, Наталія & Venger, Vitalii & Sokolov, Volodymyr. (2024). Tokenomics and Perspectives of Proof of Stake.
28. Ivanichenko, Yevhen and Kozachok, Valerii and Dreis, Yurii and Nesterova, Olena and Dmytriienko, Kate (2021) *Exposing Deviations in Information Processes using Multifractal Analysis* Cybersecurity Providing in Information and Telecommunication Systems II 2021, 3187 (1). pp. 251-259. ISSN 1613-0073
29. Bilous, Vladyslav and Bodnenko, Dmitry and Lokaziuk, Oleksandra and Skladannyi, Pavlo and Abramov, Vadym (2025) *Cyber Evidence Software as the Digital Forensics Tools in the Investigation of Cybercrime* Cybersecurity Providing in Information and Telecommunication Systems 2025 (3991). pp. 26-37. ISSN 1613-0073
30. Brzhevskaya, Zoreslava and Haidur, Halyna and Dovzhenko, Nadiia and Anosov, Andrii and Vorokhob, Maksym (2021) *Recurrent Estimation of the Information State Vector and the Correlation of Measuring Impact Matrix using a Multi-Agent Model* Cybersecurity Providing in Information and Telecommunication Systems 2021, 2963. pp. 272-276. ISSN 1613-0073
31. Anakhov, P. and Zhebka, V. and Kravchenko, V. and Skladannyi, Pavlo and Zhylytsov, Oleksii (2023) *Hybrid Study of the Geophysical Situation in the Depressed Zone of Hydroelectric Power Plants Reservoir* Cybersecurity Providing in Information and Telecommunication Systems, 3550. pp. 269-274. ISSN 1613-0073
32. Dovzhenko, Nadiia and Barabash, Oleg and Musienko, Andrii and Ivanichenko, Yevhen and Krasheninnik, Iryna (2024) *Enhancing Sensor Network Efficiency Through Optimized Flooding Mechanism* CPITS-2024: Cybersecurity Providing in Information and Telecommunication Systems, February 28, 2024, Kyiv, Ukraine. pp. 465-470. ISSN 1613-0073
33. Obushnyi, Sergiy and Virovets, Denis and Zhytar, Maksym and Zhdanova, Yuliia (2024) *Unlocking the Economic Potential of Real Estate Tokenization in Ukraine* Digital Economy Concepts and Technologies Workshop 2024, 3665. pp. 79-87. ISSN 1613-0073
34. Trofymchuk, Oleksandr and Vasyanin, Volodymyr and Sokolov, Volodymyr and Chikrii, Arkadii and Ushakova, Liudmyla (2021) *On the Use of Gray Codes for Solving 0-1 Combinatorial Problems of Optimization and Decision-Making in Environmental and Economic Systems* Information Technology and Mathematical Modeling for Environmental Safety 2021, 3021. pp. 18-28. ISSN 1613-0073
35. Tkachenko, O. and Tkachenko, K. and Tkachenko, O. and Kyrychok, Roman and Yaskevych, Vladyslav (2024) *Neural Networks in the Processing of Natural Language Texts in Information Learning*

Systems Cybersecurity Providing in Information and Telecommunication Systems, 3654. pp. 73-87. ISSN 1613-0073

36. Popereshnyak, S. and Ovcharenko, V. and Novikov, Y. and Hulak, Hennadii (2024) *Detection of intrusions based on text analysis and machine learning methods in the development of information systems* Cybersecurity Providing in Information and Telecommunication Systems II 2024, 3826. pp. 310-318. ISSN 1613-0073

37. Tykholaz, D. and Banakh, R. and Mychuda, L. and Piskozub, A. and Kyrychok, Roman (2024) *Incident response with AWS detective controls* Cybersecurity Providing in Information and Telecommunication Systems II 2024, 3826. pp. 190-197. ISSN 1613-0073

38. Shevchenko, Svitlana & Zhdanova, Yuliia & Nehodenko, Olena & Spasiteleva, Svitlana & Nehodenko, Vitalii. (2025). *Research of Information Conflict between Humans and Artificial Intelligence in Information and Cybernetic Systems*

39. Avdeyenko, Gleb and Narytnyk, Teodor and Saiko, Volodymyr and Kriuchkova, Larysa and Tsmokanych, Ivan (2025) *Terahertz Range Transceiver Module for Electromagnetic Protection of Objects* Cybersecurity Providing in Information and Telecommunication Systems 2025 (3991). pp. 603-610. ISSN 1613-0073

40. Hulak, H., et al. (2020). *Cryptovirology: Security Threats to Guaranteed Information Systems and Measures to Combat Encryption Viruses*. Cybersecurity: Education, Science, Technique, 2(10), 6–28. <https://doi.org/10.28925/2663-4023.2020.10.628>

41. Sokolov, V. (2022). *Approaches to the formation of scientific thinking in cybersecurity high school students*. Cybersecurity: Education, Science, Technique, 2(18), 124–137. <https://doi.org/10.28925/2663-4023.2022.18.124137>

42. Shevchenko, S., Skladannyi, P., Nehodenko, O., & Nehodenko, V. (2022). *Study of applied aspects of conflict theory in security systems*. Cybersecurity: Education, Science, Technique, 2(18), 150–162. <https://doi.org/10.28925/2663-4023.2022.18.150162>

43. Sukaylo, I., & Korshun, N. (2022). *The influence of NLU and generative AI on the development of cyber defense systems*. Cybersecurity: Education, Science, Technique, 2(18), 187–196. <https://doi.org/10.28925/2663-4023.2022.18.187196>

44. Костюк Ю., Бебешко Б., Гулак Г., Складаний П., Рзаєва С., Хорольська К. (2024) *Забезпечення кібербезпеки та швидкодії передачі даних у безпроводних мережах*. Безпека інформації, т. 30, №3, 365-375.

45. Hulak, H., Zhdanova, Y., Skladannyi, P., Hulak, Y., & Korniiets, V. (2022). Vulnerabilities of short message encryption in mobile information and communication systems of critical infrastructure objects. *Cybersecurity: Education, Science, Technique*, 1(17), 145–158. <https://doi.org/10.28925/2663-4023.2022.17.145158>
46. Romaniuk, O., Skladannyi, P., & Shevchenko, S. (2022). Comparative analysis of solutions to provide control and management of privileged access in the IT environment. *Cybersecurity: Education, Science, Technique*, 4(16), 98–112. <https://doi.org/10.28925/2663-4023.2022.16.98112>
47. Borkivska, O., Zhurakovskiy, B., & Platonenko, A. (2022). Smart city information system based on LoRa technology. *Cybersecurity: Education, Science, Technique*, 4(16), 113–128. <https://doi.org/10.28925/2663-4023.2022.16.113128>
48. Brzhevska, Z., Kyrychok, R., Platonenko, A., & Hulak, H. (2022). Assessment of the preconditions of formation of the methodology of assessment of information reliability. *Cybersecurity: Education, Science, Technique*, 3(15), 164–174. <https://doi.org/10.28925/2663-4023.2022.15.164174>
49. Shevchenko, S., Zhdanova, Y., Skladannyi, P., & Boiko, S. (2022). Insiders and insider information: Essence, threats, activities and legal responsibility. *Cybersecurity: Education, Science, Technique*, 3(15), 175–185. <https://doi.org/10.28925/2663-4023.2022.15.175185>
50. Довженко, Н., Мазур, Н., Костюк, Ю., Рзаєва, С. Інтеграція ІОТ ТА штучного інтелекту в інтелектуальні транспортні системи. *Кібербезпека: освіта, наука, техніка*. 2024. 2(26), 430–444.
51. Bogachuk, V. Sokolov and V. Buriachok, "Monitoring Subsystem for Wireless Systems Based on Miniature Spectrum Analyzers," *2018 International Scientific-Practical Conference Problems of Infocommunications. Science and Technology (PIC S&T)*, Kharkiv, Ukraine, 2018, pp. 581-585, doi: 10.1109/INFOCOMMST.2018.8632151.
52. Hulak, Hennadii and Skladannyi, Pavlo and Sokolov, Volodymyr and Hulak, E. and Korniiets, V. (2022) *Dynamic model of guarantee capacity and cyber security management in the critical automated systems* 2nd International Conference on Conflict Management in Global Information Networks (CMiGiN 2022), 3530. pp. 102-111. ISSN 1613-0073

КОНТРОЛЬНІ ПИТАННЯ

1. Які математичні галузі лежать в основі сучасної криптографії і чому їх розуміння важливе для захисту даних?

2. Поясніть поняття «односторонньої» операції в криптографії і наведіть приклад задачі, яка реалізує таку властивість.
3. Що таке PRG/PRF/PRP і яка їхня роль у криптосистемах?
4. Чим відрізняються симетричні й асиметричні алгоритми за призначенням у криптографічному стеці сучасних систем?
5. Назвіть основні властивості AEAD-схем і поясніть, чому їх рекомендують використовувати в сучасних протоколах.
6. У чому полягає принцип роботи AES (включно з основними кроками раунду) і як число раундів залежить від довжини ключа?
7. Що таке GCM-режим у AES, які його переваги і які критичні вимоги до коректного застосування (зокрема щодо nonce)?
8. Порівняйте ChaCha20-Poly1305 і AES-GCM: коли та чому доцільно обирати кожен з них?
9. Опишіть архітектуру Twofish і поясніть ключовий конструктивний елемент, який відрізняє його від інших блокових шифрів.
10. Чому повторне використання одного й того ж nonce/IV з тим самим ключем є критичною помилкою для поточкових/AEAD-алгоритмів?
11. Яка операційна практика щодо генерації та зберігання ключів гарантує зниження ризику компрометації (перелічіть основні елементи lifecycle-менеджменту)?
12. Поясніть, на які загрози впливає алгоритм Гровера і як це позначається на вимогах до довжини симетричних ключів.
13. У чому суть RSA — коротко опишіть, як генерується публічний/приватний ключ та як відбувається шифрування/розшифрування.
14. Що таке ECC і чому при тих самих рівнях безпеки ECC дозволяє використовувати значно коротші ключі, ніж RSA?
15. Назвіть основні сімейства постквантових алгоритмів і поясніть основний архітектурний підхід до їх поетапного впровадження в існуючі протоколи (що таке гібридний KEX і crypto-agility).
16. Які властивості повинна мати криптографічна хеш-функція (перелічіть мінімум три) і чому MD5 та SHA-1 не рекомендовані для криптографічних застосунків?
17. Опишіть процес створення та перевірки електронного цифрового підпису (ЕЦП) у загальних рисах і чому зазвичай підписують хеш повідомлення, а не весь файл.
18. Порівняйте TLS 1.3, QUIC і WireGuard за коротким описом: яка їх роль, які ключові переваги та у яких сценаріях кожен з них застосовується?

19. Що таке IPsec (AH, ESP, SA), які режими роботи він підтримує і в яких випадках доцільно використовувати тунельний режим замість транспортного?

20. Назвіть основні виклики й практичні заходи, які треба реалізувати в організації для підготовки до постквантової ери (конкретні кроки з тестування/міграції/операцій).

СПИСОК РЕКОМЕНДОВАНИХ ДЖЕРЕЛ

1. Костюк Ю.В., Складанний П.М., Бебешко Б.Т., Хорольська К.В., Рзаєва С.Л., Ворохоб М.В. (2025) Безпека інформаційно-комунікаційних систем. Київський університет імені Бориса Грінченка, Україна, Київ. - 1017 с.
2. Костюк Ю.В., Складанний П.М., Гулак Г.М., Бебешко Б.Т., Хорольська К.В., Рзаєва С.Л. (2025) Системи захисту інформації. Київський університет імені Бориса Грінченка, Україна, Київ. - 886с.
3. Kostiuk, Yu. V., Skladannyi, P. M., Bebeshko, B. T., Khorolska, K. V., Rzaieva, S. L., & Vorokhob, M. V. (2025). Information and communication systems security. [Textbook] Kyiv: Borys Grinchenko Kyiv Metropolitan University.
4. Kostiuk, Yu. V., Skladannyi, P. M., Hulak, H. M., Bebeshko, B. T., Khorolska, K. V., & Rzaieva, S. L. (2025). Information security systems. [Textbook] Kyiv: Borys Grinchenko Kyiv Metropolitan University.
5. Гулак Г. М., Жильцов О. Б., Киричок Р. В., Коршун Н. В., Складанний П. М. Інформаційна та кібернетична безпека підприємства : підруч. / Г. М. Гулак, О. Б. Жильцов, Р. В. Киричок, Н. В. Коршун, П. М. Складанний – Львів : Видавець Марченко Т. В., 2023. – 370 с.
6. Шевченко, С., Жданова, Ю., & Гаркушенко, А. (2025). Когнітивний підхід в інформаційній та кібербезпеці. Електронне фахове наукове видання «Кібербезпека: освіта, наука, техніка», 1(29), 854–866. <https://doi.org/10.28925/2663-4023.2025.29.945>
7. Чернігівський, І., & Крючкова, Л. (2025). Тестування нейромережових моделей для вирішення задачі виявлення заражених ПК на базі цифрових слідів. Електронне фахове наукове видання «Кібербезпека: освіта, наука, техніка», 1(29), 800–817. <https://doi.org/10.28925/2663-4023.2025.29.941>
8. Романюк , В., & Платоненко, А. (2025). Аналіз якості генераторів рожевого шуму. Електронне фахове наукове видання «Кібербезпека: освіта, наука, техніка», 1(29), 789–799. <https://doi.org/10.28925/2663-4023.2025.29.940>
9. Соболенко , І., & Платоненко, А. (2025). Автоматизоване виявлення аномалій у трафіку корпоративних бездротових мереж за допомогою Python: методи, реалізація та оцінка ефективності. Електронне фахове наукове видання «Кібербезпека: освіта, наука, техніка», 1(29), 777–788. <https://doi.org/10.28925/2663-4023.2025.29.939>
10. Соколов, В. (2025). Технологія відслідковування переміщення абонентів територією підприємства критичної інфраструктури. Електронне фахове наукове видання «Кібербезпека: освіта, наука, техніка», 1(29), 207–222. <https://doi.org/10.28925/2663-4023.2025.29.920>

11. Складанний, П., Гулак, Г., & Корнієць, В. (2025). Коаліційний підхід до управління кібербезпекою інформаційних систем що застосовують хмарні технології. *Кібербезпека: освіта, наука, техніка*, 4(28), 8–25. <https://doi.org/10.28925/2663-4023.2025.27.825>
12. Довженко, Н., Іваніченко, Є., Аушева, Н., Шевчук, Ю., & Луковський, Т. (2025). Дослідження архітектури дата-центрів з інтеграцією IoT-компонентів для забезпечення енергоефективності та кіберстійкості. *Кібербезпека: освіта, наука, техніка*, 4(28), 547–564. <https://doi.org/10.28925/2663-4023.2025.28.835>
13. Методичні рекомендації до виконання курсової роботи з дисципліни «Захист інформації в інформаційно-комунікаційних системах» для студентів спеціальності 125 Кібербезпека та захист інформації, освітньої програми 125.00.01 Безпека інформаційних і комунікаційних систем / Укладачі: Костюк Ю.В., Складанний П.М, Рзаєва С.Л. Київ: КСУБГ, 2025. 70 с.
14. Шевченко, С., Жданова, Ю., Складанний, П., & Іщук, М. (2025). Створення блокчейн-платформи для електронного голосування. *Кібербезпека: освіта, наука, техніка*, 4(28), 701–714. <https://doi.org/10.28925/2663-4023.2025.28.860>
15. Крючкова, Л., & Шандрук, М. (2025). Методи протидії в радіонавігаційних конфліктах. *Кібербезпека: освіта, наука, техніка*, 4(28), 766–780. <https://doi.org/10.28925/2663-4023.2025.28.863>
16. Цехмейстер, Р., Платоненко, А., Ворохоб, М., Черевик, В., & Семеняка, С. (2025). Дослідження методів забезпечення інформаційної безпеки у віртуальному середовищі. *Кібербезпека: освіта, наука, техніка*, 3(27), 63–71. <https://doi.org/10.28925/2663-4023.2025.27.703>
17. Kostiuk, Y., Skladannyi, P., Korshun, N., Bebeshko, B., & Khorolska, K. Integrated protection strategies and adaptive resource distribution for secure video streaming over a Bluetooth network. Workshop on Cybersecurity Providing in Information and Telecommunication Systems II, (CPITS-II 2024) October 26, 2024. Aachen: CEUR, 2024. Vol. 3826. P. 129–138. ISSN: 1613-0073.
18. Олійник, Я., Платоненко, А., Черевик, В., Ворохоб, М., & Шевчук, Ю. (2025). Методи захисту інформації в технологіях IoT. *Кібербезпека: освіта, наука, техніка*, 3(27), 100–108. <https://doi.org/10.28925/2663-4023.2025.27.705>
19. Дмитрієнко, К., & Коршун, Н. (2025). Застосування вдосконалених моделей Курамото ДЛЯ ідентифікації дезінформації у соціальних мережах. *Кібербезпека: освіта, наука, техніка*, 3(27), 406–416. <https://doi.org/10.28925/2663-4023.2025.27.754>

20. Kostiuk, Y., Skladannyi, P., Samoilenko, Y., Khorolska, K., Bebeshko, B., Sokolov, V. A system for assessing the interdependencies of information system agents in information security risk management using cognitive maps. Workshop on Third International Conference on Cyber Hygiene & Conflict Management in Global Information Networks, (CH&CMiGIN'24) January 24–27, 2024. Aachen: CEUR, 2024. Vol. 3925. P. 249-264. ISSN 1613-0073.

21. Методичні рекомендації до виконання курсової роботи з дисципліни «Захист інформації в комп'ютерних системах та мережах» для студентів спеціальності 123 Комп'ютерна інженерія, освітньої програми 123.00.01 Комп'ютерна інженерія / Укладачі: Костюк Ю.В., Складанний П.М., Рзаєва С.Л. Київ: КСУБГ, 2025. 82 с.

22. Kostiuk, Y., Skladannyi, P., Sokolov, V., Hulak, H., Korshun, N. Models and algorithms for analyzing information risks during the security audit of personal data information system. Workshop on Third International Conference on Cyber Hygiene & Conflict Management in Global Information Networks, (CH&CMiGIN'24) January 24–27, 2024. Aachen: CEUR, 2024. Vol. 3925. P. 155-171. ISSN 1613-0073.

23. Довженко, Н., Складанний, П., Іваніченко, Є., & Жильцов, О. (2025). Модель динамічної взаємодії безпілотних літальних апаратів із сенсорною мережею для енергоефективного моніторингу. Кібербезпека: освіта, наука, техніка, 3(27), 466–478. <https://doi.org/10.28925/2663-4023.2025.27.766>

24. Mashkina I., Rzaieva S., Kostiuk Y., Mazur N., Brzhevskaya Z. Cybersecurity in Intelligent Transport Systems: Current Challenges and Solutions. Proceedings of the Workshop on Cybersecurity Providing in Information and Telecommunication Systems (CPITS 2025), 2025. Aachen: CEUR, 2025. Vol. 3991. P. 14-25. ISSN 1613-0073.

25. Гайдук, О., Зверев, В., & Гулак, Г. (2025). Кореляційний аналіз типів кіберінцидентів в контексті інформаційних технологій міжнародних комунікацій. Кібербезпека: освіта, наука, техніка, 3(27), 417–428. <https://doi.org/10.28925/2663-4023.2025.27.757>

26. Kostiuk Y., Skladannyi P., Sokolov V., Zhyltsov O., Ivanichenko Y. Effectiveness of Information Security Control using Audit Logs. Proceedings of the Workshop on Cybersecurity Providing in Information and Telecommunication Systems (CPITS 2025), 2025. Aachen: CEUR, 2025. Vol. 3991. P. 524-538. ISSN 1613-0073.

27. Kostiuk Y., Skladannyi, P., Sokolov, V., Vorokhob, M. Models and technologies of cognitive agents for decision-making with integration of Artificial Intelligence. Proceedings of the Modern Data Science Technologies Doctoral

Consortium (MoDaST 2025). Aachen: CEUR, 2025. Vol. 4005. P. 82–96. ISSN 1613-0073.

28. Чернігівський, І., & Крючкова, Л. (2025). Системний підхід до вирішення задачі захисту інформації в інфокомунікаційній мережі від впливу комп'ютерних вірусів. *Кібербезпека: освіта, наука, техніка*, 3(27), 572–590. <https://doi.org/10.28925/2663-4023.2025.27.781>

29. Skladannyi P., Kostiuk Y., Rzaieva S., Bebeshko, B., Korshun, N. Adaptive Methods for Embedding Digital Watermarks to Protect Audio and Video Images in Information and Communication Systems. Proceedings of the Workshop Classic, Quantum, and Post-Quantum Cryptography, August 5, 2025, Kyiv, Ukraine (CQPC 2025). Vol. 4016. P. 13–31. ISSN 1613-0073.

30. Чернігівський, І., & Крючкова, Л. (2025). Ефективні рішення для швидкого виявлення скомпрометованих ПК в інфокомунікаційних мережах. *Телекомунікаційні та інформаційні технології*, 2(87), 24–32. <https://doi.org/10.31673/2412-4338.2025.029875>

31. Kostiuk Y., Skladannyi, P., Khorolska, K., Sokolov, V., Hulak, H. Application of Statistical and Neural Network Algorithms in Steganographic Synthesis and Analysis of Hidden Information in Audio and Graphic Files. Proceedings of the Workshop Classic, Quantum, and Post-Quantum Cryptography, August 5, 2025, Kyiv, Ukraine (CQPC 2025). Vol. 4016. P. 45–65. ISSN 1613-0073.

32. Чернігівський, І., & Крючкова, Л. (2024). Тестування антивірусних рішень для корпоративного сегменту. *Безпека інформації*, 30(3), 407–413. <https://doi.org/10.18372/2225-5036.30.20362>

33. Skladannyi, P., Kostiuk Y., Khorolska, K., Bebeshko, B., Sokolov, V. Model and methodology for the formation of adaptive security profiles for the protection of wireless networks in the face of dynamic cyber threats. Proceedings of the Workshop Cyber Security and Data Protection, July 31, 2025, Lviv, Ukraine (CSDP 2025). Vol. 4042. P. 17–36. ISSN 1613-0073.

34. Соколов, В. (2025). Забезпечення стійкості безпроводових систем до атак глушіння. *Телекомунікаційні та інформаційні технології*, 1(86), 50–60. <https://doi.org/10.31673/2412-4338.2025.013623>

35. Kostiuk, Y., Rzaieva S., Khorolska, K., Mazur, N., Korshun, N. Architecture of the software system of confidential access to information resources of computer networks. Proceedings of the Workshop Cyber Security and Data Protection, July 31, 2025, Lviv, Ukraine (CSDP 2025). Vol. 4042. P. 37–53. ISSN 1613-0073.

36. Корнієць, В., & Складаний, П. (2024). Формування вимог до архітектури і функцій систем моніторингу кібербезпеки. *Телекомунікаційні*

та інформаційні технології, 4(85), 90–96. <https://doi.org/10.31673/2412-4338.2024.040224>

37. Kostiuk, Y., Skladannyi, P., Sokolov, V., Rzaieva S.(2025) Intelligent System for Simulation Modeling and Research of Information Objects. Proceedings of the 1st Workshop Software Engineering and Semantic Technologies (SEST 2025), co-located with the 15th International Scientific and Practical Programming Conference (UkrPROG 2025), May 13–14, 2025, Kyiv, Ukraine, Vol-4053, P. 237-251. ISSN 1613-0073.

38. Агашков, А., Шевченко, С., Бондарчук, А., & Жебка, В. (2024). Шляхи підвищення транспортних операцій за допомогою хмарних логістичних рішень. Телекомунікаційні та інформаційні технології, 4(85), 4–15. <https://doi.org/10.31673/2412-4338.2024.047191>

39. Sokolov, V., Kostiuk, Y., Skladannyi, P., Korshun, N. (2025). Adaptation of Network Traffic Routing Policy to Information Security and Network Protection Requirements. Proceedings of the 13th International Scientific and Practical Conference "Information Control Systems and Technologies", ICST 2025, Vol-4048, P. 397 - 411. ISSN 1613-0073.

40. Бондарчук, А., Складаний, П., Жебка, В., & Стражніков, А. (2024). Оптимізація системи зарядки електромобілів на основі методів дослідження операцій. Телекомунікаційні та інформаційні технології, 3(84), 86–93. <https://doi.org/10.31673/2412-4338.2024.038693>

41. Крючкова, Л., Башкевич, Є., & Куцибала, М. (2024). Програмна реалізація алгоритмів виокремлення контурів зображень об'єктів в інтелектуальних системах відеоспостереження. Телекомунікаційні та інформаційні технології, 2(83), 14–29. <https://doi.org/10.31673/2412-4338.2024.022439>

42. Оксанич, І., Гречанінов, В., Литвинов, В., & Складаний П. (2024). Особливості забезпечення гарантоздатності та кіберстійкості інформаційного обміну в складних умовах. Телекомунікаційні та інформаційні технології, 2(83), 105–113. <https://doi.org/10.31673/2412-4338.2024.022128>

43. Крючкова, Л., Яремчук, Д., & Пазинін, А. (2024). Планування навантаження контейнерів за допомогою лінійного програмування. Телекомунікаційні та інформаційні технології, 2(83), 114 –119. <https://doi.org/10.31673/2412-4338.2024.022935>

44. Цирканюк, Д., & Соколов, В. (2024). Методика розслідування інцидентів інформаційної безпеки. Кібербезпека: освіта, наука, техніка, 2(26), 140–154. <https://doi.org/10.28925/2663-4023.2024.26.675>

45. Козачок, В., & Драпатий, М. (2024). Аналіз технології розслідування інцидентів безпеки на об'єктах критичної інфраструктури.

Кібербезпека: освіта, наука, техніка, 2(26), 374–391. <https://doi.org/10.28925/2663-4023.2024.26.699>

46. Шевченко, С., Жданова, Ю., Складанний, П., & Петренко, Т. (2024). Нечіткі когнітивні карти як інструмент візуалізації сценаріїв реагування на інциденти в системах безпеки. Кібербезпека: освіта, наука, техніка, 2(26), 417–429. <https://doi.org/10.28925/2663-4023.2024.26.707>

47. Корнієць, В., & Жданова, Ю. (2024). Методика формування швидкісного імітостійкого шифру багато алфавітної заміни. Кібербезпека: освіта, наука, техніка, 2(26), 476–486. <https://doi.org/10.28925/2663-4023.2024.26.723>

48. Руда, Х., Сабодашко, Д., Микитин, Г., Швед, М., Бордуляк, С., & Коршун, Н. (2024). Порівняння методів цифрової обробки сигналів та моделей глибинного навчання у голосовій аутентифікації. Кібербезпека: освіта, наука, техніка, 1(25), 140–160. <https://doi.org/10.28925/2663-4023.2024.25.140160>

49. Жданова, Ю., Шевченко, С., Спасітелєва, С., & Сокульский, О. (2024). Прийняття рішень на основі лінійної оптимізації у процесі управління ризиками інформаційної безпеки. Кібербезпека: освіта, наука, техніка, 1(25), 330–343. <https://doi.org/10.28925/2663-4023.2024.25.330343>

50. Довженко, Н., Іваніченко, Є., Складанний, П., & Аушева, Н. (2024). Інтеграція безпеки та відмовостійкості сенсорних мереж на основі аналізу енергоспоживання та трафіку. Кібербезпека: освіта, наука, техніка, 1(25), 390–400. <https://doi.org/10.28925/2663-4023.2024.25.390400>

51. Іосіфов, Є., & Соколов, В. (2024). Порівняльний аналіз методів, технологій, сервісів та платформ для розпізнавання голосової інформації в системах забезпечення інформаційної безпеки. Кібербезпека: освіта, наука, техніка, 1(25), 468–486. <https://doi.org/10.28925/2663-4023.2024.25.468486>

52. Барабаш, О., Аушева, Н., Складанний, П., Іваніченко, Є., & Довженко, Н. (2024). Технічні аспекти побудови відмовостійкої інфраструктури сенсорної мережі. Кібербезпека: освіта, наука, техніка, 4(24), 185–195. <https://doi.org/10.28925/2663-4023.2024.24.185195>

53. Крючкова, Л., & Леонтьук, Н. (2024). Програмно-апаратна реалізація алгоритму швидкої оцінки потужності Wi-Fi сигналу в точках простору урбанізованого приміщення. Кібербезпека: освіта, наука, техніка, 4(24), 241–256. <https://doi.org/10.28925/2663-4023.2024.24.241256>

54. Крючкова, Л., & Дяків, В. (2024). Програмна реалізація розрахунку просторового розподілу поля рамок антен. Кібербезпека: освіта, наука, техніка, 4(24), 312–320. <https://doi.org/10.28925/2663-4023.2024.24.312320>

55. Іосіфов, Є., & Соколов, В. (2024). Методи аналізу природної мови та застосування нейронних мереж в кібербезпеці. *Кібербезпека: освіта, наука, техніка*, 4(24), 398–414. <https://doi.org/10.28925/2663-4023.2024.24.398414>
56. Шевченко, С., Жданова, Ю., Спасітелева, С., Мазур, Н., Складанний, П., & Негоденко, В. (2024). Математичні методи в кібербезпеці: кластерний аналіз та його застосування в інформаційній та кібернетичній безпеці. *Кібербезпека: освіта, наука, техніка*, 3(23), 258–273. <https://doi.org/10.28925/2663-4023.2024.23.258273>
57. Крючкова, Л., & Ємельяненко, М. (2024). Захист web-ресурсу intranet від зовнішніх і внутрішніх загроз. *Кібербезпека: освіта, наука, техніка*, 3(23), 318–327. <https://doi.org/10.28925/2663-4023.2024.23.318327>
58. Крючкова, Л., & Стеблина, О. (2024). Захист смартфонів від впливу шкідливих програм в процесі зарядки у громадських місцях. *Кібербезпека: освіта, наука, техніка*, 3(23), 338–347. <https://doi.org/10.28925/2663-4023.2024.23.338347>
59. Скіцько, О., Складанний, П., Ширшов, Р., Гуменюк, М., & Ворохоб, М. (2023). Загрози та ризики використання штучного інтелекту. *Кібербезпека: освіта, наука, техніка*, 2(22), 6–18. <https://doi.org/10.28925/2663-4023.2023.22.618>
60. Кіпчук, Ф., & Соколов, В. (2023). Модель розрахунку витрат на баг-баунті програми тестування вразливостей безпеки. *Кібербезпека: освіта, наука, техніка*, 2(22), 68–83. <https://doi.org/10.28925/2663-4023.2023.22.6883>
61. Марценюк, М., Козачок, В., Богданов, О., Іосіфов, Є., & Бржевська, З. (2023). Аналіз методів виявлення дезінформації в соціальних мережах за допомогою машинного навчання. *Кібербезпека: освіта, наука, техніка*, 2(22), 148–155. <https://doi.org/10.28925/2663-4023.2023.22.148155>
62. Машталяр, Я., Козачок, В., Бржевська, З., Богданов, О., Оксанич, І., & Литвинов, В. (2023). Дослідження розвитку та інновації кіберзахисту на об'єктах критичної інфраструктури. *Кібербезпека: освіта, наука, техніка*, 2(22), 156–167. <https://doi.org/10.28925/2663-4023.2023.22.156167>
63. Шевченко, С., Жданова, Ю., Складанний, П., & Бойко, С. (2023). Теоретико-ігровий підхід до моделювання конфліктів у системах інформаційної безпеки. *Кібербезпека: освіта, наука, техніка*, 2(22), 168–178. <https://doi.org/10.28925/2663-4023.2023.22.168178>
64. Соколов, В., & Складанний, П. (2023). Методика оцінки комплексних збитків від інциденту інформаційної безпеки. *Кібербезпека: освіта, наука, техніка*, 1(21), 99–120. <https://doi.org/10.28925/2663-4023.2023.21.99120>

65. Спасітелева, С., Чичкань, І., Шевченко, С., & Жданова, Ю. (2023). Розробка безпечних контейнерних застосунків з мікросервісною архітектурою. *Кібербезпека: освіта, наука, техніка*, 1(21), 193–210. <https://doi.org/10.28925/2663-4023.2023.21.193210>
66. Костюк Ю., Бебешко Б., Складанний П., Рзаєва С., Хорольська К. Оптимізація буфера та пріоритетів для забезпечення безпеки У Bluetooth-мережах. *Безпека інформаційних систем і технологій*. 2025. 2(8), 5-16.
67. Ворохоб, М., Киричок, Р., Яскевич, В., Добришин, Ю., & Сидоренко, С. (2023). Сучасні перспективи застосування концепції zero trust при побудові політики інформаційної безпеки підприємства. *Кібербезпека: освіта, наука, техніка*, 1(21), 223–233. <https://doi.org/10.28925/2663-4023.2023.21.223233>
68. Добришин, Ю., Сидоренко, С., & Ворохоб, М. (2023). Автоматизована система підтримки прийняття рішення щодо відновлення пошкодженого програмного забезпечення внаслідок впливу кібератак. *Кібербезпека: освіта, наука, техніка*, 4(20), 174–182. <https://doi.org/10.28925/2663-4023.2023.20.174182>
69. Соколов, В., & Складанний, П. (2023). Порівняльний аналіз стратегій побудови другого та третього рівня освітніх програм зі спеціальності 125 «Кібербезпека». *Кібербезпека: освіта, наука, техніка*, 4(20), 183–204. <https://doi.org/10.28925/2663-4023.2023.20.183204>
70. Shevchenko, S., Zhdanova, Y., & Spasiteleva, S. (2023). Mathematical methods in cybersecurity: Catastrophe theory. *Cybersecurity: Education, Science, Technique*, 3(19), 165–175. <https://doi.org/10.28925/2663-4023.2023.19.165175>
71. Sokolov, V. (2022). Approaches to the formation of scientific thinking in cybersecurity high school students. *Cybersecurity: Education, Science, Technique*, 2(18), 124–137. <https://doi.org/10.28925/2663-4023.2022.18.124137>
72. Shevchenko, S., Skladannyi, P., Nehodenko, O., & Nehodenko, V. (2022). Study of applied aspects of conflict theory in security systems. *Cybersecurity: Education, Science, Technique*, 2(18), 150–162. <https://doi.org/10.28925/2663-4023.2022.18.150162>
73. Sukaylo, I., & Korshun, N. (2022). The influence of NLU and generative AI on the development of cyber defense systems. *Cybersecurity: Education, Science, Technique*, 2(18), 187–196. <https://doi.org/10.28925/2663-4023.2022.18.187196>
74. Костюк Ю., Бебешко Б., Гулак Г., Складанний П., Рзаєва С., Хорольська К. (2024) Забезпечення кібербезпеки та швидкодії передачі даних у безпроводних мережах. *Безпека інформації*, т. 30, №3, 365-375.

75. Hulak, H., Zhdanova, Y., Skladannyi, P., Hulak, Y., & Korniiets, V. (2022). Vulnerabilities of short message encryption in mobile information and communication systems of critical infrastructure objects. *Cybersecurity: Education, Science, Technique*, 1(17), 145–158. <https://doi.org/10.28925/2663-4023.2022.17.145158>
76. Romaniuk, O., Skladannyi, P., & Shevchenko, S. (2022). Comparative analysis of solutions to provide control and management of privileged access in the IT environment. *Cybersecurity: Education, Science, Technique*, 4(16), 98–112. <https://doi.org/10.28925/2663-4023.2022.16.98112>
77. Borkivska, O., Zhurakovskiy, B., & Platonenko, A. (2022). Smart city information system based on LoRa technology. *Cybersecurity: Education, Science, Technique*, 4(16), 113–128. <https://doi.org/10.28925/2663-4023.2022.16.113128>
78. Brzhevska, Z., Kyrychok, R., Platonenko, A., & Hulak, H. (2022). Assessment of the preconditions of formation of the methodology of assessment of information reliability. *Cybersecurity: Education, Science, Technique*, 3(15), 164–174. <https://doi.org/10.28925/2663-4023.2022.15.164174>
79. Shevchenko, S., Zhdanova, Y., Skladannyi, P., & Boiko, S. (2022). Insiders and insider information: Essence, threats, activities and legal responsibility. *Cybersecurity: Education, Science, Technique*, 3(15), 175–185. <https://doi.org/10.28925/2663-4023.2022.15.175185>
80. Довженко, Н., Мазур, Н., Костюк, Ю., Рзаєва, С. Інтеграція ІОТ ТА штучного інтелекту в інтелектуальні транспортні системи. *Кібербезпека: освіта, наука, техніка*. 2024. 2(26), 430–444.
81. Bogachuk, V. Sokolov and V. Buriachok, "Monitoring Subsystem for Wireless Systems Based on Miniature Spectrum Analyzers," *2018 International Scientific-Practical Conference Problems of Infocommunications. Science and Technology (PIC S&T)*, Kharkiv, Ukraine, 2018, pp. 581-585, doi: 10.1109/INFOCOMMST.2018.8632151.
82. Hulak, Hennadii and Skladannyi, Pavlo and Sokolov, Volodymyr and Hulak, E. and Korniiets, V. (2022) *Dynamic model of guarantee capacity and cyber security management in the critical automated systems* 2nd International Conference on Conflict Management in Global Information Networks (CMiGiN 2022), 3530. pp. 102-111. ISSN 1613-0073
83. Sokolov, V. Y. and Skladannyi, Pavlo and Hulak, Hennadii (2022) *Stability Verification of Self-Organized Wireless Networks with Block Encryption* Cybersecurity Providing in Information and Telecommunication Systems, 3137. pp. 227-237. ISSN 1613-0073
84. Anakhov, P. and Zhebka, V. and Popershenyuk, S and Skladannyi, Pavlo and Sokolov, Volodymyr (2023) *Protecting Objects of Critical*

Information Infrastructure from Wartime Cyber Attacks by Decentralizing the Telecommunications Network Cybersecurity Providing in Information and Telecommunication Systems, 3550. pp. 240-245. ISSN 1613-0073

85. Костюк Ю., Довженко, Н., Мазур, Н., Складанний, П., Рзаєва, С. Методика захисту GRID-середовища від шкідливого коду під час виконання обчислювальних завдань. Кібербезпека: освіта, наука, техніка. 2025. 3(27), 22–40.

86. Kipchuk, F. and Sokolov, Volodymyr and Skladannyi, Pavlo and Zhylytsov, Oleksii and Ageyev, D. (2023) *Method for Increasing the Various Sources Data Consistency for IoT Sensors* IEEE 9th International Conference on Problems of Infocommunications, Science and Technology (PICST). pp. 522-526. ISSN 979-8-3503-9891-5

87. A. Carlsson, D. Ageyev, Y. Sadykov and V. Sokolov, "Sustainability Research of the Secure Wireless Communication System with Channel Reservation," *2020 IEEE 15th International Conference on Advanced Trends in Radioelectronics, Telecommunications and Computer Engineering (TCSET)*, Lviv-Slavske, Ukraine, 2020, pp. 973-977, doi: 10.1109/TCSET49122.2020.235583.

88. Складанний, П., Костюк Ю., Рзаєва, С., Самойленко, Ю., Савченко, Т. Розробка модульних нейронних мереж для виявлення різних класів мережевих атак. Кібербезпека: освіта, наука, техніка. 2025. 3(27), 534–548.

89. F. Kipchuk, V. Sokolov, V. Buriachok and L. Kuzmenko, "Investigation of Availability of Wireless Access Points based on Embedded Systems," *2019 IEEE International Scientific-Practical Conference Problems of Infocommunications, Science and Technology (PIC S&T)*, Kyiv, Ukraine, 2019, pp. 1-5, doi: 10.1109/PICST47496.2019.9061551.

90. Shevchuk, D and Harasymchuk, O and Partyka, A and Korshun, Natalia (2023) *Designing Secured Services for Authentication, Authorization, and Accounting of Users* Cybersecurity Providing in Information and Telecommunication Systems, 3550. pp. 207-225. ISSN 1613-0073

91. Складанний П.М., Костюк Ю.В., Мазур Н.П., Пітайчук М.А. Дослідження характеристик та продуктивності протоколів доступу до хмарних обчислювальних середовищ на основі універсального тестування. Телекомунікаційні та інформаційні технології. 2025. №1(86). С. 61-74.

92. Kuzminykh, I., Ghita, B., Sokolov, V., & Bakhshi, T. (2021). Information Security Risk Assessment. *Encyclopedia*, 1(3), 602-617. <https://doi.org/10.3390/encyclopedia1030050>

93. TajDini, M., Sokolov, V., Kuzminykh, I., Shiaeles, S., & Ghita, B. (2020). Wireless Sensors for Brain Activity—A Survey. *Electronics*, 9(12), 2092. <https://doi.org/10.3390/electronics9122092>

94. Grechaninov, Viktor and Hulak, Hennadii and Hulak, Evgen and Skladannyi, Pavlo and Sokolov, Volodymyr (2021) *Decentralized Access Demarcation System Construction in Situational Center Network Cybersecurity Providing in Information and Telecommunication Systems II* 2021, 3188 (2). pp. 197-206. ISSN 1613-0073

95. Buriachok, Volodymyr and Ageyev, Dmytro and Zhylytsov, Oleksii and Skladannyi, Pavlo and Sokolov, V. Y. (2020) *Invasion Detection Model using Two-Stage Criterion of Detection of Network Anomalies* Cybersecurity Providing in Information and Telecommunication Systems, 2746. pp. 23-32. ISSN 1613-0073

96. Костюк Ю., Хорольська, К., Бебешко, Б., Довженко, Н., Коршун, Н., Пазинін, А. Інструментальні засоби забезпечення інформаційної безпеки від прихованих загроз в інфраструктурі хмарних обчислень. Кібербезпека: освіта, наука, техніка. 2025. 4 (28), 633–655.

97. M. Vladymyrenko, V. Sokolov, V. Buriachok, A. Platonenko and D. Ageyev, "Analysis of Implementation Results of the Distributed Access Control System," *2019 IEEE International Scientific-Practical Conference Problems of Infocommunications, Science and Technology (PIC S&T)*, Kyiv, Ukraine, 2019, pp. 1-6, doi: 10.1109/PICST47496.2019.9061376.

98. Складанний, П.М., Машкіна, І.В., Рзаєва, С.Л., Костюк, Ю.В. Методи GDPR для забезпечення безпеки сховищ даних від витоків та загроз. Телекомунікаційні та інформаційні технології. 2025. (2), 59-76.

99. Hu, Z., Buriachok, V., Bogachuk, I., Sokolov, V., Ageyev, D. (2021). Development and Operation Analysis of Spectrum Monitoring Subsystem 2.4–2.5 GHz Range. In: Radivilova, T., Ageyev, D., Kryvinska, N. (eds) *Data-Centric Business and Applications. Lecture Notes on Data Engineering and Communications Technologies*, vol 48. Springer, Cham. https://doi.org/10.1007/978-3-030-43070-2_29

100. F. Kipchuk, V. Sokolov, P. Skladannyi and D. Ageyev, "Assessing Approaches of IT Infrastructure Audit," *2021 IEEE 8th International Conference on Problems of Infocommunications, Science and Technology (PIC S&T)*, Kharkiv, Ukraine, 2021, pp. 213-217, doi: 10.1109/PICST54195.2021.9772181.

101. Grechaninov, Viktor and Hulak, Hennadii and Sokolov, V. Y. and Skladannyi, Pavlo and Korshun, Natalia (2022) *Formation of Dependability and Cyber Protection Model in Information Systems of Situational Center* Emerging Technology Trends on the Smart Industry and the Internet of Things 2022, 3149. pp. 107-117. ISSN 1613-0073

102. Костюк, Ю., Складанний, П., Рзаєва, С., Мазур, Н., Черевик, В., & Аносов, А. (2025). Особливості реалізації мережових атак через TCP/IP-протоколи. Електронне фахове наукове видання «Кібербезпека: освіта, наука, техніка», 1(29), 571–597.
103. Kuzminykh, I., Carlsson, A., Yevdokymenko, M., Sokolov, V. (2019). Investigation of the IoT Device Lifetime with Secure Data Transmission. In: Galinina, O., Andreev, S., Balandin, S., Koucheryavy, Y. (eds) Internet of Things, Smart Spaces, and Next Generation Networks and Systems. NEW2AN ruSMART 2019 2019. Lecture Notes in Computer Science(), vol 11660. Springer, Cham. https://doi.org/10.1007/978-3-030-30859-9_2
104. Buriachok, Volodymyr and Sokolov, V. Y. and Skladannyi, Pavlo (2019) *Security Rating Metrics for Distributed Wireless Systems* MoMLeT&DS. pp. 222-233. ISSN 1613-0073
105. Hu, Z.B., Buriachok, V., TajDini, M., Sokolov, V. (2021). Authentication System by Human Brainwaves Using Machine Learning and Artificial Intelligence. In: Hu, Z., Petoukhov, S., Dychka, I., He, M. (eds) Advances in Computer Science for Engineering and Education IV. ICCSEEA 2021. Lecture Notes on Data Engineering and Communications Technologies, vol 83. Springer, Cham. https://doi.org/10.1007/978-3-030-80472-5_31
106. Костюк Ю., Бебешко, Б., Крючкова, Л., Литвинов, В., Оксанич, І., Складанний, П., Хорольська, К. Захист інформації та безпека обміну даними в безпроводових мобільних мережах з автентифікацією і протоколами обміну ключами. Кібербезпека: освіта, наука, техніка. 2024. 1(25), 229–252.
107. Buhas, Vasyl and Ponomarenko, Ihor and Bugas, Valeriy and Ramskyi, Andrii and Sokolov, Volodymyr (2021) *Using Machine Learning Techniques to Increase the Effectiveness of Cybersecurity* Cybersecurity Providing in Information and Telecommunication Systems II 2021, 3188 (2). pp. 273-281. ISSN 1613-0073
108. Zhebka, V. and Skladannyi, Pavlo and Zhebka, S. and Shlianachak, S. and Bondarchuk, A. (2024) *Methodology for Predicting Failures in a Smart Home based on Machine Learning Methods* Cybersecurity Providing in Information and Telecommunication Systems 2024, 3654. pp. 322-332. ISSN 1613-0073
109. Skladannyi, Pavlo and Trofimov, Olexandr and Korniiets, Viktor and Vorokhob, Maksym and Opryshko, Tetiana (2023) *Improving the Security Policy of the Distance Learning System based on the Zero Trust Concept* CPITS 2023: Workshop on Cybersecurity Providing in Information and Telecommunication Systems. pp. 97-106. ISSN 1613-0073

110. Довженко, Н., Іваніченко, Є., & Костюк, Ю. (2025). Методика виявлення та локалізації кіберзагроз у хмарних середовищах з інтегрованими ІОТ-компонентами на основі графових моделей. Електронне фахове наукове видання «Кібербезпека: освіта, наука, техніка», 1(29), 762–776.
111. Шевченко, Галина Володимирівна and Шевченко, Світлана Миколаївна and Жданова, Юлія Дмитрівна and Спасітелєва, Світлана Олексіївна and Негоденко, Олена Василівна (2021) *Information Security Risk Analysis SWOT Cybersecurity Providing in Information and Telecommunication Systems*, 2923. pp. 309-317. ISSN 1613-0073
112. I. Iosifov, O. Iosifova and V. Sokolov, "Sentence Segmentation from Unformatted Text using Language Modeling and Sequence Labeling Approaches," *2020 IEEE International Conference on Problems of Infocommunications. Science and Technology (PIC S&T)*, Kharkiv, Ukraine, 2020, pp. 335-337, doi: 10.1109/PICST51311.2020.9468084.
113. Petriv, P. and Opirskyy, I. and Mazur, Nataliia (2024) *Modern technologies of decentralized databases, authentication, and authorization methods Cybersecurity Providing in Information and Telecommunication Systems II* 2024, 3826. pp. 60-71. ISSN 1613-0073
114. Astafieva, Mariia and Bodnenko, Dmitry and Lytvyn, Oksana and Proshkin, Volodymyr and Skladannyi, Pavlo (2023) *Formation of High School Students' Resistance to Destructive Information Influences Cybersecurity Providing in Information and Telecommunication Systems* 2023, 3421. pp. 87-96. ISSN 1613-0073
115. Sovyn, Y. and Khoma, V. and Opirskyy, I. and Kozachok, Valerii (2023) *Minimization of Bitsliced Representation of 4×4 S-Boxes based on Ternary Logic Instruction Cybersecurity Providing in Information and Telecommunication Systems* 2023, 3421. pp. 12-24. ISSN 1613-0073
116. Burov, O. et al. (2020). Cybersecurity in Educational Networks. In: Ahram, T., Karwowski, W., Vergnano, A., Leali, F., Taiar, R. (eds) *Intelligent Human Systems Integration 2020. IHSI 2020. Advances in Intelligent Systems and Computing*, vol 1131. Springer, Cham. https://doi.org/10.1007/978-3-030-39512-4_56
117. Buriachok, V., Sokolov, V. (2020). Implementation of Active Learning in the Master's Program on Cybersecurity. In: Hu, Z., Petoukhov, S., Dychka, I., He, M. (eds) *Advances in Computer Science for Engineering and Education II. ICCSEEA 2019. Advances in Intelligent Systems and Computing*, vol 938. Springer, Cham. https://doi.org/10.1007/978-3-030-16621-2_57
118. Anakhov, Pavlo and Zhebka, Viktoriia and Koretska, Viktoriia and Sokolov, V. Y. and Skladannyi, Pavlo (2022) *Increasing the*

Functional Network Stability in the Depression Zone of the Hydroelectric Power Station Reservoir Emerging Technology Trends on the Smart Industry and the Internet of Things, 3149. pp. 169-176. ISSN 1613-0073

119. Z. Hu, B. Vovkotrub, V. Buriachok, Y. Zotkin and V. Sokolov, "Bandwidth Research of Wireless IoT Switches," *2020 IEEE 15th International Conference on Advanced Trends in Radioelectronics, Telecommunications and Computer Engineering (TCSET)*, Lviv-Slavske, Ukraine, 2020, pp. 1-5, doi: 10.1109/TCSET49122.2020.235492.

120. Bessalov, Anatoly and Sokolov, Volodymyr and Skladannyi, Pavlo and Abramov, Serhii and Zhylytsov, Oleksii (2022) *Modeling CSIKE Algorithm on Non-Cyclic Edwards Curves* Cybersecurity Providing in Information and Telecommunication Systems 2022, 3288 (1). pp. 1-10. ISSN 1613-0073

121. Bessalov, Anatoly and Sokolov, Volodymyr and Skladannyi, Pavlo and Mazur, Nataliia and Ageyev, Dmytro (2021) *Implementation of the CSIDH Algorithm Model on Supersingular Twisted and Quadratic Edwards Curves* Cybersecurity Providing in Information and Telecommunication Systems II 2021, 3187 (1). pp. 302-309. ISSN 1613-0073

122. Bessalov, Anatoly and Kovalchuk, L. and Sokolov, V. Y. and Skladannyi, Pavlo and Radivilova, T. (2020) *Analysis of 2-Isogeny Properties of Generalized Form Edwards Curves* Cybersecurity Providing in Information and Telecommunication Systems, 2746. pp. 1-13. ISSN 1613-0073

123. Bessalov, Anatolii & Grubiyan, Evgeniy & Sokolov, Volodymyr & Skladannyi, Pavlo. (2020). 3- and 5-Isogenies of Supersingular Edwards Curves. 10.48550/arXiv.2006.16111.

124. Bessalov, Anatoly and Sokolov, V. Y. and Skladannyi, Pavlo and Zhylytsov, Oleksii (2021) *Computing of Odd Degree Isogenies on Supersingular Twisted Edwards Curves* Cybersecurity Providing in Information and Telecommunication Systems, 2923. pp. 1-11. ISSN 1613-0073

125. V. Astapenya, V. Sokolov, P. Skladannyi and O. Zhylytsov, "Analysis of Ways and Methods of Increasing the Availability of Information in Distributed Information Systems," *2021 IEEE 8th International Conference on Problems of Infocommunications, Science and Technology (PIC S&T)*, Kharkiv, Ukraine, 2021, pp. 174-178, doi: 10.1109/PICST54195.2021.9772161.

126. V. Astapenya, V. Buriachok, V. Sokolov, P. Skladannyi and D. Ageyev, "Last Mile Technique for Wireless Delivery System using an Accelerating Lens," *2020 IEEE International Conference on Problems of Infocommunications. Science and Technology (PIC S&T)*, Kharkiv, Ukraine, 2020, pp. 811-814, doi: 10.1109/PICST51311.2020.9467886.

127. Bessalov, Anatoly and Abramov, Serhii and Sokolov, Volodymyr and Skladannyi, Pavlo and Zhylytsov, Oleksii (2023) *Multifunctional CRS Encryption Scheme on Isogenies of Non-Supersingular Edwards Curves* Workshop on Classic, Quantum, and Post-Quantum Cryptography (CQPC 2023), 3504. pp. 12-25. ISSN 1613-0073
128. TajDini, Mahyar and Sokolov, V. Y. and Kuzminykh, Ievgeniia and Ghita, B. (2023) *Brainwave-based authentication using features fusion* Computers & Security (129). p. 103198. ISSN 0167-4048
129. Iosifova, Olena and Iosifov, Ievgen and Rolik, Oleksandr and Sokolov, V. Y. (2020) *Techniques Comparison for Natural Language Processing* MoMLet&DS, 2631 (I). pp. 57-67. ISSN 1613-0073
130. Anakhov, P. and Zhebka, V. and Tushych, A. and Kravchenko, V. and Blazhennyi, N. and Skladannyi, Pavlo and Sokolov, Volodymyr (2022) *Evaluation Method of the Physical Compatibility of Equipment in a Hybrid Information Transmission Network* Journal of Theoretical and Applied Information Technology, 100 (22). pp. 6635-6644. ISSN 1817-3195
131. Y. Sadykov, V. Sokolov and P. Skladannyi, "Technology of Location Hiding by Spoofing the Mobile Operator IP Address," *2021 IEEE International Conference on Information and Telecommunication Technologies and Radio Electronics (UkrMiCo)*, Odesa, Ukraine, 2021, pp. 22-25, doi: 10.1109/UkrMiCo52950.2021.9716700.
132. Obushnyi, Sergiy and Virovets, Denis and Hulak, Hennadii and Zhurakovskiy, Bohdan (2022) *Autonomy of Economic Agents in Peer-to-Peer Systems* Cybersecurity Providing in Information and Telecommunication Systems 2022, 3288 (1). pp. 125-133. ISSN 1613-0073
133. TajDini, Mahyar and Sokolov, V. Y. and Buriachok, Volodymyr (2019) *Men-in-the-Middle Attack Simulation on Low Energy Wireless Devices using Software Define Radio* MoMLet&DS (2386). pp. 287-296. ISSN 1613-0073
134. V. Sokolov, P. Skladannyi and A. Platonenko, "Video Channel Suppression Method of Unmanned Aerial Vehicles," *2022 IEEE 41st International Conference on Electronics and Nanotechnology (ELNANO)*, Kyiv, Ukraine, 2022, pp. 473-477, doi: 10.1109/ELNANO54667.2022.9927105.
135. Bessalov, Anatoliy and Abramov, Serhii and Sokolov, Volodymyr and Mazur, Nataliia (2023) *CSIKE-ENC Combined Encryption Scheme with Optimized Degrees of Isogeny Distribution* Cybersecurity Providing in Information and Telecommunication Systems, 3421. pp. 36-45. ISSN 1613-0073
136. Hulak, Hennadii and Kriuchkova, Larysa and Skladannyi, Pavlo and Opirskyy, Ivan (2021) *Formation of Requirements for the Electronic*

Record-Book in Guaranteed Information Systems of Distance Learning Cybersecurity Providing in Information and Telecommunication Systems 2021, 2963. pp. 137-142. ISSN 1613-0073

137. Iosifova, Olena and Iosifov, Ievgen and Sokolov, V. Y. and Romanovskyi, O. and Sukaylo, I. (2021) *Analysis of Automatic Speech Recognition Methods Cybersecurity Providing in Information and Telecommunication Systems*, 2923. pp. 252-257. ISSN 1613-0073

138. Zhebka, V. and Zhebka, S and Bazhan, T and Skladannyi, Pavlo and Sokolov, Volodymyr (2024) *Methodology for Choosing a Consensus Algorithm for Blockchain Technology Digital Economy Concepts and Technologies Workshop* 2024, 3665. pp. 106-113. ISSN 1613-0073

139. Dudykevych, V. and Mykytyn, G. and Stosyk, T. and Skladannyi, Pavlo (2024) *Platform for the Security of Cyber-Physical Systems and the IoT in the Intellectualization of Society Cybersecurity Providing in Information and Telecommunication Systems* 2024, 3654. pp. 449-457. ISSN 1613-0073

140. Shevchenko, Svitlana & Zhdanova, Yuliia & Shevchenko, Halyna & Nehodenko, Olena & Spasiteleva, Svitlana. (2023). *Information Security Risk Management using Cognitive Modeling*.

141. V. Sokolov, P. Skladannyi and A. Platonenko, "Jump-Stay Jamming Attack on Wi-Fi Systems," *2023 IEEE 18th International Conference on Computer Science and Information Technologies (CSIT)*, Lviv, Ukraine, 2023, pp. 1-5, doi: 10.1109/CSIT61576.2023.10324031.

142. Buriachok, V., Korshun, N., Zhylytsov, O., Sokolov, V., Skladannyi, P. (2023). *Implementation of Active Cybersecurity Education in Ukrainian Higher School*. In: Faure, E., Danchenko, O., Bondarenko, M., Tryus, Y., Bazilo, C., Zaspá, G. (eds) *Information Technology for Education, Science, and Technics. ITEST 2022. Lecture Notes on Data Engineering and Communications Technologies*, vol 178. Springer, Cham. https://doi.org/10.1007/978-3-031-35467-0_32

143. Marusenko, R., Sokolov, V., Buriachok, V. (2021). *Experimental Evaluation of Phishing Attack on High School Students*. In: Hu, Z., Petoukhov, S., Dychka, I., He, M. (eds) *Advances in Computer Science for Engineering and Education III. ICCSEEA 2020. Advances in Intelligent Systems and Computing*, vol 1247. Springer, Cham. https://doi.org/10.1007/978-3-030-55506-1_59

144. V. Astapenya, V. Sokolov and D. Ageyev, "Experimental Evaluation of an Accelerating Lens on Spatial Field Structure and Frequency Spectrum," *2020 IEEE Ukrainian Microwave Week (UkrMW)*, Kharkiv, Ukraine, 2020, pp. 203-206, doi: 10.1109/UkrMW49653.2020.9252755.

145. Brzhevská, Zoreslava and Kyrychok, Roman and Anosov, Andrii and Skladannyi, Pavlo and Vorokhob, Maksym (2021) *Analysis of the*

Process of Information Transfer from the Source-to-User in Terms of Information Impact Cybersecurity Providing in Information and Telecommunication Systems II 2021, 3188 (2). pp. 257-264. ISSN 1613-0073

146. HU, ZHENG BING & Buriachok, Volodymyr & Sokolov, Volodymyr. (2019). Implementation of Social Engineering Attack at Institution of Higher Education. SSRN Electronic Journal. 10.2139/ssrn.3679106.

147. Mykhaylova, O and Fedynyshyn, T and Sokolov, Volodymyr and Kyrychok, Roman (2024) *Person-of-Interest Detection on Mobile Forensics Data—AI-Driven Roadmap* Cybersecurity Providing in Information and Telecommunication Systems 2024, 3654. pp. 239-251. ISSN 1613-0073

148. Marusenko, R., Sokolov, V., Skladannyi, P. (2023). Social Engineering Penetration Testing in Higher Education Institutions. In: Hu, Z., Dychka, I., He, M. (eds) *Advances in Computer Science for Engineering and Education VI. ICCSEEA 2023. Lecture Notes on Data Engineering and Communications Technologies*, vol 181. Springer, Cham. https://doi.org/10.1007/978-3-031-36118-0_96

149. Mykhaylova, O and Fedynyshyn, T and Datsiuk, A and Fihol, B and Hulak, Hennadii (2023) *Mobile Application as a Critical Infrastructure Cyberattack Surface* Cybersecurity Providing in Information and Telecommunication Systems II 2023, 3050. pp. 29-43. ISSN 1613-0073

150. Kyrychok, R., Laptiev, O., Lisnevskyi, R., Kozlovskyi, V., & Klobukov, V. (2022). Development of a method for checking vulnerabilities of a corporate network using Bernstein transformations. *Eastern-European Journal of Enterprise Technologies*, 1(9(115)), 93–101. <https://doi.org/10.15587/1729-4061.2022.253530>

151. V. Sokolov, P. Skladannyi and V. Astapenya, "Bluetooth Low-Energy Beacon Resistance to Jamming Attack," *2023 IEEE 13th International Conference on Electronics and Information Technologies (ELIT)*, Lviv, Ukraine, 2023, pp. 270-274, doi: 10.1109/ELIT61488.2023.10310815.

152. Astapenya, Volodymyr & Zhdanova, Yuliia & Shevchenko, Svitlana & Spasiteleva, Svitlana & Kryvoruchko, Olena. (2024). Conflict Model of Radio Engineering Systems under the Threat of Electronic Warfare.

153. Dudykevych, V and Yevseiev, S and Mykytyn, H and Ruda, K and Hulak, Hennadii (2024) *Detecting Deepfake Modifications of Biometric Images using Neural Networks* Cybersecurity Providing in Information and Telecommunication Systems 2024, 3654. pp. 391-397. ISSN 1613-0073

154. Kostiuk, Yuliia. (2025). Multi-Agent System for Detecting and Counteracting Attacks on the Enterprise Information System. In book: *Insider*

Threats and Security in Corporations. pp.205-232.
<https://doi.org/10.36690/ITSC-205-232>.

155. Adamantis, M., Sokolov, V., Skladannyi, P. (2025). Evaluation of State-of-the-Art Machine Learning Smart Contract Vulnerability Detection Method. In: Hu, Z., Yanovsky, F., Dychka, I., He, M. (eds) Advances in Computer Science for Engineering and Education VII. ICCSEEA 2024 2024. Lecture Notes on Data Engineering and Communications Technologies, vol 242. Springer, Cham. https://doi.org/10.1007/978-3-031-84228-3_5

156. Chernenko, Roman and Anosov, Andrii and Kyrychok, Roman and Brzhevskaya, Zoreslava and Spasiteleva, Svitlana (2022) *Encryption Method for Systems with Limited Computing Resources* Cybersecurity Providing in Information and Telecommunication Systems 2022, 3288 (1). pp. 142-148. ISSN 1613-0073

157. Iavich, M. and Kovalchuk, O. and Gnatyuk, S. and Khavikova, Y. and Sokolov, Volodymyr (2024) *Classical and post-quantum encryption for GDPR* Classic, Quantum, and Post-Quantum Cryptography 2024, 3829. pp. 70-78. ISSN 1613-0073

158. Romanovskiy, O. and Iosifov, Ievgen and Iosifova, Olena and Sokolov, Volodymyr and Skladannyi, Pavlo and Sukaylo, Igor (2022) *Prototyping Methodology of End-to-End Speech Analytics Software* Modern Machine Learning Technologies and Data Science Workshop, 3312 (1). pp. 76-86. ISSN 1613-0073

159. Romanovskiy, O., Iosifov, I., Iosifova, O., Sokolov, V., Kipchuk, F., Sukaylo, I. (2021). Automated Pipeline for Training Dataset Creation from Unlabeled Audios for Automatic Speech Recognition. In: Hu, Z., Petoukhov, S., Dychka, I., He, M. (eds) Advances in Computer Science for Engineering and Education IV. ICCSEEA 2021. Lecture Notes on Data Engineering and Communications Technologies, vol 83. Springer, Cham. https://doi.org/10.1007/978-3-030-80472-5_3

160. Bessalov, A., Sokolov, V., & Abramov, S. (2024). Efficient Commutative PQC Algorithms on Isogenies of Edwards Curves. *Cryptography*, 8(3), 38. <https://doi.org/10.3390/cryptography8030038>

161. Syrotynskiy, R. and Tyshyk, I. and Kochan, O. and Sokolov, Volodymyr and Skladannyi, Pavlo (2024) *Methodology of network infrastructure analysis as part of migration to zero-trust architecture* Cyber Security and Data Protection 2024 (3800). pp. 97-105. ISSN 1613-0073

162. Martseniuk, Y. and Partyka, A. and Harasymchuk, O. and Shevchenko, Svitlana (2024) *Universal centralized secret data management for automated public cloud provisioning* Cybersecurity Providing in

Information and Telecommunication Systems II 2024, 3826. pp. 72-81. ISSN 1613-0073

163. Iosifov, Ievgen and Iosifova, Olena and Sokolov, Volodymyr and Skladannyi, Pavlo and Sukaylo, Igor (2021) *Natural Language Technology to Ensure the Safety of Speech Information Cybersecurity Providing in Information and Telecommunication Systems II*. Volume I, 3187 (1). pp. 216-226. ISSN 1613-0073

164. Bahatskyi, Oleksii and Bahatskyi, V. and Sokolov, Volodymyr (2023) *Smart Home Subsystem for Calculating the Quality of Public Utilities Cybersecurity Providing in Information and Telecommunication Systems*, 3421. pp. 168-173. ISSN 1613-0073

165. V. Sokolov, P. Skladannyi and N. Mazur, "Wi-Fi Repeater Influence on Wireless Access," 2023 *IEEE 5th International Conference on Advanced Information and Communication Technologies (AICT)*, Lviv, Ukraine, 2023, pp. 33-36, doi: 10.1109/AICT61584.2023.10452421.

166. Dreis, Y. and Korchenko, O. and Brzhevska, Zoreslava and Kriuchkova, Larysa and Nesterova, Olena (2024) *Model to Formation Data Base of Internal Parameters for Assessing the Status of the State Secret Protection Cybersecurity Providing in Information and Telecommunication Systems* 2024, 3654. pp. 277-289. ISSN 1613-0073

167. Virovets, Denis and Obushnyi, Sergiy and Zhurakovskiy, B. and Skladannyi, Pavlo and Sokolov, Volodymyr (2024) *Integration of smart contracts and artificial intelligence using cryptographic oracles* Classic, Quantum, and Post-Quantum Cryptography 2024, 3829. pp. 39-46. ISSN 1613-0073

168. Shevchenko, Svitlana & Zhdanova, Yuliia & Shevchenko, Halyna & Nehodenko, Olena & Spasiteleva, Svitlana. (2023). Conflict Analysis in the "Subject-to-Subject" Security System.

169. Myrontsov, Mykyta and Noskov, Oleksii and Sokolov, Volodymyr (2021) *Method for Determining Areas of Stable Solutions of Inverse Problems of Mathematical Physics by the Example of Well Electrometry* Information Technology and Mathematical Modeling for Environmental Safety 2021, 3021. pp. 115-126. ISSN 1613-0073

170. Buriachok, V., Hadzhyiev, M., Sokolov, V., Skladannyi, P., & Kuzmenko, L. (2019). Implantation of indexing optimization technology for highly specialized terms based on Metaphone phonetical algorithm. *Eastern-European Journal of Enterprise Technologies*, 5(2 (101), 43–50. <https://doi.org/10.15587/1729-4061.2019.181943>

171. Kostiuk, Y. V., Skladannyi, P. M., Bebeshko, B. T., Khorolska, K. V., Rzaieva, S. L., & Vorokhob, M. V. (2025). Information and communication

systems security.[Textbook] Kyiv: Borys Grinchenko Kyiv Metropolitan University.

172. Dreis, Yurii and Ivanichenko, Yevhen and Nesterova, Olena and Zhdanova, Yuliia and Dmytriienko, Kate (2022) *Restricted Information Identification Model* Cybersecurity Providing in Information and Telecommunication Systems 2022, 3288 (1). pp. 89-95. ISSN 1613-0073

173. Martseniuk, Yevhenii and Partyka, Andrii and Harasymchuk, Oleh and Cherevyk, Vyacheslav and Dovzhenko, Nadiia (2025) *Research of the Centralized Configuration Repository Efficiency for Secure Cloud Service Infrastructure Management* Cybersecurity Providing in Information and Telecommunication Systems 2025 (3991). pp. 260-274. ISSN 1613-0073

174. Gnatyuk, S. and Sydorenko, V. and Polozhentsev, A. and Sokolov, Volodymyr (2024) *Method for managing IT incidents in critical information infrastructure facilities* Cybersecurity Providing in Information and Telecommunication Systems II 2024, 3826. pp. 326-333. ISSN 1613-0073

175. Shapoval, V. and Zubyk, L. and Zubyk, Y. and Kurchenko, O. and Kozachok, Valerii (2024) *Automation of Data Management Processes in Cloud Storage* Cybersecurity Providing in Information and Telecommunication Systems 2024, 3654. pp. 410-418. ISSN 1613-0073

176. Dovzhenko, N. and Haidur, H. and Brzhevskaya, Zoreslava and Ivanichenko, Yevhen and Nesterova, Olena (2023) *Method of Sensor Network Functioning under the Redistribution Condition of Requests between Nodes* Cybersecurity Providing in Information and Telecommunication Systems 2023, 3421. pp. 278-283. ISSN 1613-0073

177. Kulibaba, Serhii and Popereshnyak, Svitlana and Shcheblanin, Yuri and Kurchenko, Oleg and Mazur, Nataliia (2022) *Advanced Communication Model with the Voice Control and the Increased Security Level* Cybersecurity Providing in Information and Telecommunication Systems 2022, 3288 (1). pp. 64-72. ISSN 1613-0073

178. Zybin, Serhii and Khoroshko, Volodymyr and Khokhlachova, Yuliia and Kozachok, Valerii (2021) *Approach of the Attack Analysis to Reduce Omissions in the Risk Management* Cybersecurity Providing in Information and Telecommunication Systems 2021, 2963. pp. 318-328. ISSN 1613-0073

179. V. Sokolov, P. Skladannyi and V. Astapenya, "Wi-Fi Interference Resistance to Jamming Attack," 2023 *IEEE 5th International Conference on Advanced Information and Communication Technologies (AICT)*, Lviv, Ukraine, 2023, pp. 1-4, doi: 10.1109/AICT61584.2023.10452687.

180. Shevchenko, Svitlana and Zhdanova, Yuliia and Kryvytska, O. and Shevchenko, H. and Spasiteleva, Svitlana (2024) *Fuzzy cognitive mapping as a scenario approach for information security risk*

analysis Cybersecurity Providing in Information and Telecommunication Systems II 2024, 3826. pp. 356-362. ISSN 1613-0073

181. Pinchuk, Olga and Sokolyuk, Oleksandra and Burov, Oleksandr and Lavrov, Evgeniy and Shevchenko, Svitlana and Aksakovska, Valeriia (2020) *ICT for training and evaluation of the solar impact on aviation safety* ICT in Education, Research, and Industrial Applications. Proc. 16 th Int. Conf. ICTERI 2020. Volume II: Workshops. Kharkiv, Ukraine, October 6-10, 2020, CEUR-WS.org., pp. 786-792. ISSN 1613-0073

182. Zhurakovskiy, B. and Otrokh, S. and Poliakov, M. and Poliakov, O. and Skladannyi, Pavlo (2024) *Enhancing information transmission security with stochastic codes* Classic, Quantum, and Post-Quantum Cryptography 2024, 3829. pp. 62-69. ISSN 1613-0073

183. Kriuchkova, Larysa and Tsmokanych, Ivan and Shevchenko, Svitlana and Bohdanov, Oleksandr and Mazur, Nataliia (2023) *Experimental Research of the Parameters of Danger and Protective Signals Attached to High-Frequency Imposition* Cybersecurity Providing in Information and Telecommunication Systems, 3550. pp. 261-268. ISSN 1613-0073

184. Berestov, Denis and Kurchenko, Oleg and Zubyk, L. and Kulibaba, Serhii and Mazur, Nataliia (2023) *Assessment of Weather Risks for Agriculture using Big Data and Industrial Internet of Things Technologies* Cybersecurity Providing in Information and Telecommunication Systems II 2023, 3550. pp. 1-13. ISSN 1613-0073

185. Tkachenko, K. and Tkachenko, O. and Tkachenko, O. and Mazur, Nataliia and Mashkina, Iryna (2024) *Ontological Approach in Modern Educational Processes* Cybersecurity Providing in Information and Telecommunication Systems 2024, 3654. pp. 88-97. ISSN 1613-0073

186. Popereshnyak, S. and Novikov, Y. and Zhdanova, Yuliia (2024) *Cryptographic system security approaches by monitoring the random numbers generation* Cybersecurity Providing in Information and Telecommunication Systems II 2024, 3826. pp. 301-309. ISSN 1613-0073

187. Mykhaylova, O. and Stefankiv, A. and Nakonechny, T. and Fedynyshyn, T. and Sokolov, Volodymyr (2024) *Resistance to Replay Attacks of Remote Control Protocols using the 433 MHz Radio Channel* Cybersecurity Providing in Information and Telecommunication Systems 2024, 3654. pp. 98-110. ISSN 1613-0073

188. Novytskyi, Anton and Sokolov, Volodymyr and Kriuchkova, Larysa and Skladannyi, Pavlo (2025) *Determining the error distribution of BLE beacons at antenna near and far fields* Cyber Hygiene & Conflict Management in Global Information Networks 2025, 4024. pp. 133-143. ISSN 1613-0073

189. Kostiuk, Y. V., Skladannyi, P. M., Hulak, H. M., Bebesko, B. T., Khorolska, K. V., & Rzaieva, S. L. (2025). Information security systems.[Textbook] Kyiv: Borys Grinchenko Kyiv Metropolitan University.

190. Romanovskyi, O. *et al.* (2025). Accuracy Improvement of Spoken Language Identification System for Close-Related Languages. In: Hu, Z., Yanovsky, F., Dychka, I., He, M. (eds) *Advances in Computer Science for Engineering and Education VII. ICCSEE 2024* 2024. Lecture Notes on Data Engineering and Communications Technologies, vol 242. Springer, Cham. https://doi.org/10.1007/978-3-031-84228-3_4

191. Основи баз даних : навчальний посібник / Рзаєва С.Л., Машкіна І.В., Складаний П.М., Костюк Ю.В., Рзаєв Д.О., Красюк Ю.М – Київ : Київський столичний університет імені Бориса Грінченка, 2025. – 319 с.

192. Kriuchkova, Larysa and Tsmokanych, I. and Vovk, M. and Mazur, Nataliia and Bohdanov, Oleksandr (2024) *Influence of Protective Signals on Dangerous Signals of High-Frequency Imposition* Cybersecurity Providing in Information and Telecommunication Systems 2024, 3654. pp. 419-425. ISSN 1613-0073

193. Buhas, V. and Ponomarenko, I. and Buhas, N. and Hulak, Hennadii (2024) *Cybersecurity Role in AI-Powered Digital Marketing* Digital Economy Concepts and Technologies Workshop 2024, 3665. pp. 1-11. ISSN 1613-0073

194. Dreis, Y. and Korchenko, O. and Sokolov, Volodymyr and Skladannyi, Pavlo (2024) *Model to formation data base of secondary parameters for assessing status of the state secret protection* Cyber Security and Data Protection 2024 (3800). pp. 1-11. ISSN 1613-0073

195. Shevchenko, Svitlana & Zhdanova, Yuliia & Astapenya, Volodymyr & Nehodenko, Olena & Spasiteleva, Svitlana. (2024). *Conflicting Subsystems in the Information Space: A Study at the Software and Hardware Levels.*

196. Kolbasova, K and Zhurakovskiy, Bohdan and Poltorak, V and Nakonechnyi, V and Kyrychok, Roman (2023) *Smart Home Network based on Cisco Equipment* Cybersecurity Providing in Information and Telecommunication Systems, 3550. pp. 70-80. ISSN 1613-0073

197. Virovets, Denis and Obushnyi, Sergiy and Ramskyi, Andrii and Hulak, Hennadii (2024) *Soul Bound Token as Digital Twins in Peer-to-Peer Economic Systems* Digital Economy Concepts and Technologies Workshop 2024, 3665. pp. 97-105. ISSN 1613-0073

198. Tyshyk, I. and Hulak, Hennadii (2024) *Testing an organization's information system for unauthorized access* Cybersecurity Providing in Information and Telecommunication Systems II 2024, 3826. pp. 17-29. ISSN 1613-0073

199. Zhebka, Serhii and Zhebka, Viktoriia and Hulak, Hennadii and Kyrychok, Roman and Platonenko, Artem (2025) *Method for Adaptive Allocation of Cryptographic Resources in Distributed Databases* Cybersecurity Providing in Information and Telecommunication Systems 2025 (3991). pp. 620-628. ISSN 1613-0073
200. Dovzhenko, Nadiia & Ivanichenko, Yevhen & Skladannyi, Pavlo & Zhyltsov, Oleksii. (2024). Research of UAV and sensor network integration features for routing optimization and energy consumption reduction.
201. Zhebka, V. and Skladannyi, Pavlo and Bazak, Y and Bondarchuk, A. and Storchak, K. (2024) *Methods for Predicting Failures in a Smart Home* Digital Economy Concepts and Technologies Workshop 2024, 3665. pp. 70-78. ISSN 1613-0073
202. Rzaeva S., Skladannyi P., Kostiuk Y., Abramov V., Kravchenko V. Adaptive Information Security Management in Cloud-Oriented Intelligent Transportation Systems. Ukrainian Scientific Journal of Information Security, 2025, vol. 31, issue 1, pp. 23-36. <https://doi.org/10.18372/2225-5036.31.20634>
203. Kriuchkova, Larysa and Tsmokanych, I. and Mazur, Nataliia and Tarasenko, D. and Osadcha, V. (2024) *Experimental determination of protective signal parameters for effective “swinging” of the carrier frequency of high-frequency imposition* Cybersecurity Providing in Information and Telecommunication Systems II 2024, 3826. pp. 251-259. ISSN 1613-0073
204. Kraus, N. and Kraus, K. and Shevchenko, Svitlana and Proshkin, Volodymyr (2024) *Digitalizing Production and Industry through 4.0 Technologies Implementation* Digital Economy Concepts and Technologies Workshop 2024, 3665. pp. 36-47. ISSN 1613-0073
205. Kulibaba, S. and Zubyk, L. and Kurchenko, O. and Sinitsyn, I. and Mazur, Nataliia (2023) *Linked System of Data Organization and Management* Cybersecurity Providing in Information and Telecommunication Systems (3421). pp. 117-126. ISSN 1613-0073
206. Chyzhevskia, Maryna & Романовська, Наталія & Venger, Vitalii & Sokolov, Volodymyr. (2024). Tokenomics and Perspectives of Proof of Stake.
207. Ivanichenko, Yevhen and Kozachok, Valerii and Dreis, Yurii and Nesterova, Olena and Dmytriienko, Kate (2021) *Exposing Deviations in Information Processes using Multifractal Analysis* Cybersecurity Providing in Information and Telecommunication Systems II 2021, 3187 (1). pp. 251-259. ISSN 1613-0073
208. Bilous, Vladyslav and Bodnenko, Dmitry and Lokaziuk, Oleksandra and Skladannyi, Pavlo and Abramov, Vadym (2025) *Cyber Evidence Software as the Digital Forensics Tools in the Investigation of*

Cybercrime Cybersecurity Providing in Information and Telecommunication Systems 2025 (3991). pp. 26-37. ISSN 1613-0073

209. Brzhevska, Zoreslava and Haidur, Halyna and Dovzhenko, Nadiia and Anosov, Andrii and Vorokhob, Maksym (2021) *Recurrent Estimation of the Information State Vector and the Correlation of Measuring Impact Matrix using a Multi-Agent Model* Cybersecurity Providing in Information and Telecommunication Systems 2021, 2963. pp. 272-276. ISSN 1613-0073

210. Anakhov, P. and Zhebka, V. and Kravchenko, V. and Skladannyi, Pavlo and Zhytsov, Oleksii (2023) *Hybrid Study of the Geophysical Situation in the Depressed Zone of Hydroelectric Power Plants Reservoir* Cybersecurity Providing in Information and Telecommunication Systems, 3550. pp. 269-274. ISSN 1613-0073

211. Dovzhenko, Nadiia and Barabash, Oleg and Musienko, Andrii and Ivanichenko, Yevhen and Krashenninnik, Iryna (2024) *Enhancing Sensor Network Efficiency Through Optimized Flooding Mechanism* CPITS-2024: Cybersecurity Providing in Information and Telecommunication Systems, February 28, 2024, Kyiv, Ukraine. pp. 465-470. ISSN 1613-0073

212. Obushnyi, Sergiy and Virovets, Denis and Zhytar, Maksym and Zhdanova, Yuliia (2024) *Unlocking the Economic Potential of Real Estate Tokenization in Ukraine* Digital Economy Concepts and Technologies Workshop 2024, 3665. pp. 79-87. ISSN 1613-0073

213. Складанний, П.М., Гулак, Г.М., Костюк, Ю.В. Генератор хаотичних чисел із нечітким керуванням для криптографічних систем із динамічною довірою. Телекомунікаційні та інформаційні технології. 2025. № 4 (89), 137-147. <https://doi.org/10.31673/2412-4338.2025.048916>

214. Trofymchuk, Oleksandr and Vasyanin, Volodymyr and Sokolov, Volodymyr and Chikrii, Arkadii and Ushakova, Liudmyla (2021) *On the Use of Gray Codes for Solving 0-1 Combinatorial Problems of Optimization and Decision-Making in Environmental and Economic Systems* Information Technology and Mathematical Modeling for Environmental Safety 2021, 3021. pp. 18-28. ISSN 1613-0073

215. Рзаєва С.Л., Складанний П.М., Машкіна І.В., Костюк Ю.В. (2025) Модель реалізації керування доступом на основі ролей (RBAC) у багаторівневій архітектурі сховища даних. Сучасний захист інформації, 3(63), 137–149.

216. Костюк, Ю., Складанний, П., Рзаєва, С., Самойленко, Ю., & Коршун, Н. (2025). Інтелектуальні системи керування та захисту в кіберфізичних і хмарних середовищах Smart Grid. Електронне фахове наукове видання «Кібербезпека: освіта, наука, техніка», 2(30), 125–156.

217. Tkachenko, O. and Tkachenko, K. and Tkachenko, O. and Kyrychok, Roman and Yaskevych, Vladyslav (2024) *Neural Networks in the Processing of Natural Language Texts in Information Learning Systems* Cybersecurity Providing in Information and Telecommunication Systems, 3654. pp. 73-87. ISSN 1613-0073
218. Popereshnyak, S. and Ovcharenko, V. and Novikov, Y. and Hulak, Hennadii (2024) *Detection of intrusions based on text analysis and machine learning methods in the development of information systems* Cybersecurity Providing in Information and Telecommunication Systems II 2024, 3826. pp. 310-318. ISSN 1613-0073
219. Складаний, П., Костюк, Ю., Рзаєва, С., & Мазур, Н. (2025). *Паралельна обробка даних у розширюваних хеш-структурах та оцінка їх продуктивності*. Електронне фахове наукове видання «Кібербезпека: освіта, наука, техніка», 3(31), 242–269. <https://doi.org/10.28925/2663-4023.2025.31.1015>
220. Tykholaz, D. and Banakh, R. and Mychuda, L. and Piskozub, A. and Kyrychok, Roman (2024) *Incident response with AWS detective controls* Cybersecurity Providing in Information and Telecommunication Systems II 2024, 3826. pp. 190-197. ISSN 1613-0073
221. Shevchenko, Svitlana & Zhdanova, Yuliia & Nehodenko, Olena & Spasiteleva, Svitlana & Nehodenko, Vitalii. (2025). *Research of Information Conflict between Humans and Artificial Intelligence in Information and Cybernetic Systems*
222. Avdeyenko, Gleb and Narytnyk, Teodor and Saiko, Volodymyr and Kriuchkova, Larysa and Tsmokanych, Ivan (2025) *Terahertz Range Transceiver Module for Electromagnetic Protection of Objects* Cybersecurity Providing in Information and Telecommunication Systems 2025 (3991). pp. 603-610. ISSN 1613-0073
223. Hulak, H., et al. (2020). *Cryptovirology: Security Threats to Guaranteed Information Systems and Measures to Combat Encryption Viruses*. Cybersecurity: Education, Science, Technique, 2(10), 6–28. <https://doi.org/10.28925/2663-4023.2020.10.628>

Інтернет-джерела

1. Рекомендації CERT-UA щодо протидії APT (Advanced Persistent Threat). URL: <https://cert.gov.ua/recommendations/18>

2. Технологія VPN може загрожувати кібербезпеці. URL: <https://detector.media/infospace/article/126505/2017-05-31-tekhnologiya-vpnmozhe-zagrozhuvati-kiberbezpetsi-inau>
3. JP 3–12 Cyberspace Operations, 8 June 2018. URL: https://www.jcs.mil/Portals/36/Documents/Doctrine/pubs/jp3_12.pdf
4. DOD Dictionary of Military and Associated Terms. As of January 2019. URL: <https://www.jcs.mil/Portals/36/Documents/Doctrine/pubs/dictionary.pdf>
5. Cybersecurity a generic reference curriculum (2016). URL: https://www.nato.int/nato_static_fl2014/assets/pdf/pdf_2016_10/1610-cybersecurity-162_curriculum.pdf
6. National Cyber Security Strategy Good Practice Guide (2016). URL: <https://www.enisa.europa.eu/publications/ncss-good-practice-guide>
7. Castro D. (2018). Boosting the Cyberworkforce. URL: <http://www.govtech.com/data/Boosting-the-Cyberworkforce.html>
8. Cisco -Україна. URL: <https://www.cisco.com>
9. Про основні засади забезпечення кібербезпеки України: Закон України // Відомості Верховної Ради України. 2017. № 45. Ст. 403.
10. Закон України "Про інформацію".
11. Закон України "Про захист інформації в інформаційно-комунікаційних системах".
12. ДСТУ ISO/IEC 27001:2015 (ISO/IEC 27001:2013, IDT) "Інформаційні технології. Методи захисту. Системи управління інформаційною безпекою. Вимоги".
13. ДСТУ ISO/IEC 27002:2015 "Інформаційні технології. Методи захисту. Практичні правила управління інформаційною безпекою".
14. ДСТУ ISO/IEC 27005:2022 "Інформаційні технології. Методи захисту. Управління ризиками інформаційної безпеки".
15. ДСТУ ISO/IEC 27701:2022 "Розширення для управління конфіденційністю в системах управління інформаційною безпекою".
16. ДСТУ ISO/IEC 15408:2020 "Критерії оцінювання безпеки інформаційних технологій".
17. ДСТУ ISO/IEC 24762:2015 "Інформаційні технології. Методи захисту. Рекомендації з відновлення інформаційно-комунікаційних технологій після інцидентів".

Навчальне видання

КОСТЮК Юлія Володимирівна,
СКЛАДАННИЙ Павло Миколайович

ЗАХИСТ ІНФОРМАЦІЇ В КОМП'ЮТЕРНИХ СИСТЕМАХ ТА МЕРЕЖАХ

Підручник

*Частина I. Теоретичні та методологічні основи
захисту інформації в комп'ютерних системах та
мережах*