Київський столичний університет імені Бориса Грінченка
Факультет інформаційних технологій та математики
Кафедра інформаційної та кібернетичної безпеки
імені професора Володимира Бурячка

«Допущено до захисту»
Завідувач кафедри інформаційної та кібернетичної безпеки імені професора Володимира Бурячка
кандидат технічних наук, доцент
Складанний П.М.

_____
(підпис)
« ___ » _____ 2025 р.

КВАЛІФІКАЦІЙНА РОБОТА
на здобуття другого (магістерського)
рівня вищої освіти

Спеціальність 125 Кібербезпека та захист інформації

Тема роботи:
ДОСЛІДЖЕННЯ МЕТОДІВ ТА РОЗРОБКА РЕКОМЕНДАЦІЙ ЩОДО ЗАСТОСУВАННЯ ЗАСОБІВ ЗАХИСТУ ІНФОРМАЦІЇ В ХМАРНОМУ СЕРЕДОВИЩІ

Виконав

студент групи БІКСм-1-25-1.4д

Боровський Максим Ігорович

(прізвище, ім’я, по батькові)

Науковий керівник
Кандидат військових наук, доцент
(науковий ступінь, наукове звання)

Аносов Андрій Олександрович
(прізвище, ініціали)

Київ – 2025

Київський столичний університет імені Бориса Грінченка
Факультет інформаційних технологій та математики
Кафедра інформаційної та кібернетичної безпеки
імені професора Володимира Бурячка

Освітньо-кваліфікаційних рівень – магістр
Спеціальність 125 Кібербезпека та захист інформації
Освітня програма 125.00.01 Безпека інформаційних і комунікаційних систем

«Затверджую»
Завідувач кафедри інформаційної та кібернетичної безпеки імені професора Володимира Бурячка
кандидат технічних наук, доцент
Складанний П.М.

_____
(підпис)
« » 2025 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ
Боровському Максиму Ігоровичу
(прізвище, ім'я, по батькові)

1. Тема роботи: Дослідження методів та розробка рекомендацій щодо застосування засобів захисту інформації в хмарному середовищі;
   керівник к.в.н., доц. Аносов Андрій Олександрович
   затверджені наказом ректора від «___»_____ 20___ року №__.
2. Термін подання студентом роботи «___»_____ 20___ р.
3. Вихідні дані до роботи:
   3.1 науково-технічна та нормативна література з теми дослідження: науково-технічні праці - 55, ISO/IEC 27001:2022, ISO/IEC 27005:2022, ISO/IEC 22301:2019, NIST SP 800-61 Rev.2, NIST SP 800-207, NIST SP 800-137, Закон України «Про основні засади забезпечення кібербезпеки України», Закон України «Про інформацію», Постанова КМУ №518 від 19.06.2019;
   3.2 методи: системний аналіз, моделювання загроз в хмарному середовищі, структурне та функціональне проєктування, ризик-орієнтовані методи оцінювання загроз, експериментальні методи тестування механізмів автентифікації й авторизації;
   3.3 технології: CIS Benchmarks, NIST SP 800-53, CSA Cloud Controls Matrix, AWS Security Hub Best Practices;
4. Зміст текстової частини роботи (перелік питань, які потрібно розробити):
   4.1. Провести аналіз сучасних проблем і загроз інформаційній безпеці в хмарних середовищах на основі наукових і галузевих джерел.
   4.2. Дослідити існуючі методи, стандарти та фреймворки захисту (CIS Benchmarks, NIST SP 800-53, CSA Cloud Controls Matrix, AWS Security Hub Best Practices тощо).
   4.3. Визначити недоліки наявних рекомендацій та методик оцінювання безпеки.
   4.4. Сформулювати систему критеріїв (показників) для оцінювання ефективності застосування засобів захисту в хмарному середовищі.

4.5. Розробити методичні рекомендації щодо підвищення рівня захищеності хмарних ресурсів.

4.6. Створити прототип програмного засобу (утиліту) для автоматизованої перевірки конфігурацій AWS-акаунтів і демонстрації ефективності запропонованих підходів.

5. Перелік графічного матеріалу:

5.1 Презентація доповіді, виконана в Microsoft PowerPoint.

6. Дата видачі завдання «___»_____ 20___ р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів підготовки роботи	Термін виконання	Примітка
1.	Уточнення постановки завдання		
2.	Аналіз літератури		
3.	Обґрунтування вибору рішення		
4.	Збір даних		
5.	Виконання та оформлення розділу 1.		
6.	Виконання та оформлення розділу 2.		
7.	Виконання та оформлення розділу 3.		
8.	Вступ, висновки, реферат		
9.	Апробація роботи на науково-методичному семінарі та/або науково-технічній конференції		
10.	Оформлення та друк текстової частини роботи		
11.	Оформлення презентацій		
12.	Отримання рецензій		
13.	Попередній захист роботи		
14.	Захист в ЕК		

Студент _____ Боровський Максим Ігорович
(підпис) (прізвище, ім’я, по батькові)

Науковий керівник _____ Аносов Андрій Олександрович
(підпис) (прізвище, ім’я, по батькові)

РЕФЕРАТ

У роботі досліджено актуальні проблеми та загрози інформаційній безпеці у хмарних середовищах, зокрема в екосистемі Amazon Web Services (AWS). Визначено, що понад 60% інцидентів безпеки спричинені помилками в конфігураціях, надлишковими привілеями та відсутністю належного моніторингу. Проаналізовано ключові стандарти та фреймворки (CIS, NIST, CSA), на основі яких розроблено систему критеріїв оцінки захищеності та методичні рекомендації. Практична цінність дослідження полягає у створенні програмного засобу (утиліти) на мові Python, що дозволяє автоматизувати перевірку стану безпеки облікових записів і ресурсів AWS. Підтверджено гіпотезу, що поєднання існуючих стандартів з інструментами автоматизації суттєво знижує ризик виникнення критичних уразливостей та підвищує рівень відповідності нормативним вимогам. Робота складається зі вступу, чотирьох розділів, висновків, списку використаних джерел, додатків.

Ключові слова: ХМАРНІ ТЕХНОЛОГІЇ, ІНФОРМАЦІЙНА БЕЗПЕКА, AMAZON WEB SERVICES (AWS), КІБЕРБЕЗПЕКА, АВТОМАТИЗАЦІЯ МОНІТОРИНГУ, КОНФІГУРАЦІЯ БЕЗПЕКИ, CIS BENCHMARKS, УПРАВЛІННЯ РИЗИКАМИ, ХМАРНА ІНФРАСТРУКТУРА.

Зміст

ВСТУП

Актуальність теми

Високий темп розвитку хмарних технологій викликав трансформацію підходів до зберігання, обробки та захисту інформації. Публічні та змішані хмарні середовища, такі як Amazon Web Services (AWS), Microsoft Azure та Google Cloud Platform (GCP), забезпечують організаціям високу масштабованість, гнучкість і економічну ефективність. Також з'являються нові виклики по захисту даних. Це все може бути викликано через більш складну архітектуру, нерозуміння ризиків і складні підходи до проектування безпечних. Також зі збільшенням динаміки зміни середовищ збільшуються ризики недостатнього моніторингу та проблеми з виявленням загроз.

За даними Cloud Security Alliance понад 60% проблем з безпекою пов'язані не з вразливістю платформ, а з неправильними налаштуваннями, надлишковими привілеями для користувачів і співробітників та з відсутністю моніторингу безпеки і інцидентів. Все це свідчить що є потреба в вдосконаленні інструментів контролю і моніторингу інструментів та автоматизації, що дозволяють напряму впливати на рівень безпеки і зменшувати кількість і якість ризиків. Актуальність дослідження базується на систематизації актуальних знань і ризиків в хмарних середовищах задля того щоб ці рекомендації можна було перенести на практичну площину і налаштувати автоматичний контроль конфігурацій

Мета дослідження

Метою роботи є дослідження сучасних методів забезпечення безпеки інформації в хмарному середовищі та розробка практичних рекомендацій і програмного засобу для автоматизованої перевірки стану безпеки облікових записів і ресурсів у середовищі AWS.

Завдання дослідження

Для досягнення поставленої мети необхідно вирішити такі основні завдання:

1. Провести аналіз сучасних проблем і загроз інформаційній безпеці в хмарних середовищах на основі наукових і галузевих джерел.
2. Дослідити існуючі методи, стандарти та фреймворки захисту (CIS Benchmarks, NIST SP 800-53, CSA Cloud Controls Matrix, AWS Security Hub Best Practices тощо).
3. Визначити недоліки наявних рекомендацій та методик оцінювання безпеки.
4. Сформулювати систему критеріїв (показників) для оцінювання ефективності застосування засобів захисту в хмарному середовищі.
5. Розробити методичні рекомендації щодо підвищення рівня захищеності хмарних ресурсів.
6. Створити прототип програмного засобу (утиліту) для автоматизованої перевірки конфігурацій AWS-акаунтів і демонстрації ефективності запропонованих підходів.

Об’єкт і предмет дослідження

- Об’єкт дослідження - процеси забезпечення інформаційної безпеки в хмарних обчислювальних середовищах.
- Предмет дослідження - методи, засоби та критерії оцінювання ефективності застосування заходів захисту інформації в хмарних інфраструктурах, зокрема в екосистемі Amazon Web Services.

Методи дослідження

У роботі застосовуються такі методи:

- Аналіз і синтез - для вивчення існуючих методів і стандартів безпеки.
- Порівняльний аналіз - для зіставлення різних підходів до оцінювання безпеки (CIS, NIST, CSA, ISO).
- Системний підхід - для узагальнення загроз і формування комплексних рекомендацій.
- Моделювання та експеримент - для перевірки запропонованих рішень на тестовому середовищі AWS.
- Методи автоматизованого аналізу конфігурацій - реалізовані в розробленому програмному засобі з використанням Python-бібліотек boto3, pydantic, typer, structlog.

Гіпотеза дослідження

Передбачається, що комбінування існуючих фреймворків безпеки (CIS, NIST, CSA) з автоматизованими перевірками конфігурацій хмарних ресурсів дозволить знизити кількість критичних уразливостей, скоротити час виявлення помилок безпеки та підвищити рівень відповідності нормативним вимогам.

Огляд літературних джерел

Проблеми захисту інформації в хмарних середовищах широко висвітлені у працях як українських, так і зарубіжних дослідників.

У звітах Cloud Security Alliance (CSA, 2024) визначено найактуальніші загрози - misconfiguration, data breaches, insecure interfaces, insider threats. У документі CIS Amazon Web Services Foundations Benchmark (2024) наведено базові рекомендації щодо безпеки акаунтів і сервісів AWS. NIST SP 800-53 Rev. 5 (2020) та ISO/IEC 27017:2021 визначають універсальні принципи побудови систем захисту інформації та управління ризиками. Наукові публікації (Alharthi et al., 2023; Hussain et al., 2021) досліджують систематизацію хмарних місконфігурацій і пропонують автоматизовані методи їх виявлення.

Практичні огляди (IBM X-Force, 2023; ENISA, 2023) підкреслюють зростання ролі Cloud Security Posture Management (CSPM) як інструменту постійного моніторингу хмарної безпеки.

У сукупності ці джерела формують теоретичну й методологічну основу для проведення даного дослідження, однак більшість із них не дає інтегрованого підходу до оцінювання ефективності рекомендацій, що й визначає наукову новизну та практичну значущість даної роботи.

РОЗДІЛ 1. АНАЛІЗ ПРОБЛЕМ І СУЧАСНИХ ПІДХОДІВ ДО ЗАХИСТУ ІНФОРМАЦІЇ В ХМАРНОМУ СЕРЕДОВИЩІ

1.1 Загальні засади безпеки в хмарних обчисленнях

1.1.1. Поняття та характеристики хмарних обчислень (IaaS, PaaS, SaaS)

Хмарні обчислення - це модель надання обчислювальних ресурсів як сервісу через мережу [1, 2]. Вона забезпечує користувачеві доступ до апаратних, програмних і мережевих ресурсів без необхідності володіти або обслуговувати фізичну інфраструктуру. Згідно з визначенням Національного інституту стандартів і технологій США, хмарні обчислення є моделлю, яка забезпечує зручний, постійний та на вимогу доступ до спільного пулу налаштовуваних обчислювальних ресурсів. Це можуть бути спільні мережі, сервери, сховища, додатки і сервіси, що можуть швидко надаватись та звільнятись із мінімальними управлінськими зусиллями чи взаємодією з провайдером.

Основними характеристиками хмарних обчислень є [3]:

- Самообслуговування на вимогу (on-demand self-service): користувач може самостійно створювати, налаштовувати й видаляти ресурси без втручання адміністратора постачальника.
- Ширококанальний мережевий доступ (broad network access): ресурси доступні через стандартні механізми (веб-інтерфейси, API) із будь-якого місця.
- Мультиоренда ресурсів (resource pooling): постачальник об’єднує ресурси для обслуговування багатьох клієнтів за моделлю мультиоренди (multi-tenancy).
- Різка еластичність (rapid elasticity): ресурси можуть динамічно масштабуватися залежно від потреб користувача.
- Вимірюваність послуг (measured service): використання ресурсів контролюється та оплачується відповідно до фактичного споживання.

Залежно від рівня абстракції та контролю над інфраструктурою, розрізняють три основні моделі надання хмарних послуг:

1. Інфраструктура як сервіс (Infrastructure as a Service, IaaS).

Ця модель передбачає надання користувачу базових інфраструктурних компонентів - віртуальних серверів, сховищ даних, мережевих ресурсів та систем управління. Користувач самостійно керує операційною системою, застосунками та політиками безпеки. Провайдер відповідає за фізичну інфраструктуру та її доступність. Приклади: Amazon EC2, Microsoft Azure Virtual Machines, Google Compute Engine.

2. Платформа як сервіс (Platform as a Service, PaaS).

У цій моделі користувач отримує середовище для розробки, тестування й розгортання застосунків без необхідності керувати інфраструктурою. Провайдер

надає готові інструменти (середовища виконання, бази даних, фреймворки), а користувач зосереджується на логіці застосунку і процесі розроблення. Приклади: AWS Elastic Beanstalk, Google App Engine, Microsoft Azure App Service.

3. Програмне забезпечення як сервіс (Software as a Service, SaaS).

Це модель, за якої користувач отримує повноцінний прикладний сервіс через веб сайт або API, не керуючи жодною частиною інфраструктури чи платформи. Провайдер повністю відповідає за підтримку, оновлення, безпеку та масштабування. Приклади: Google Workspace, Microsoft 365, Salesforce.

Кожна модель має власний баланс між контролем, відповідальністю та безпекою.

У IaaS користувач має повний контроль, але й максимальну відповідальність за безпеку конфігурацій (мережі, ОС, ідентичності), безпека на рівні внутрішніх сервісів все рівно регулюється провайдером. У PaaS відповідальність частково розділяється: провайдер забезпечує безпечне середовище виконання, тоді як користувач відповідає за безпеку коду, контейнерів і політики доступу. У SaaS користувач відповідає лише за налаштування облікових записів і контроль доступу до даних.

Таким чином, модель спільної відповідальності (Shared Responsibility Model) є ключовою концепцією хмарної безпеки. Вона визначає межу між обов’язками провайдера (захист інфраструктури, мереж, гіпервізора) та користувача (захист даних, ідентичностей, конфігурацій). Невірне розуміння та неправильне використання цієї моделі часто стає причиною інцидентів безпеки в хмарних середовищах.

1.1.2. Особливості архітектури та моделі розподілу відповідальності між провайдером і користувачем

Архітектура хмарних обчислень побудована на багаторівневій структурі, яка поєднує апаратні, програмні та віртуалізаційні компоненти. Основними архітектурними рівнями хмарного середовища є: фізична інфраструктура, рівень віртуалізації, сервіси платформи та рівень користувацьких застосунків.

Таке розшарування дозволяє досягти високої масштабованості, гнучкості управління ресурсами та ефективного розподілу навантажень між користувачами.

1. Фізичний рівень (Infrastructure Layer) - включає сервери, сховища даних, мережеве обладнання, системи охолодження та енергопостачання. Безпека цього рівня забезпечується постачальником хмарних послуг і включає фізичний контроль доступу, резервування та моніторинг.
2. Рівень віртуалізації (Virtualization Layer) - забезпечує логічне розділення фізичних ресурсів за допомогою гіпервізора або контейнерної платформи. Цей рівень є критичним з точки зору безпеки, адже ізоляція між

користувачами повинна бути гарантованою, щоб уникнути несанкціонованого доступу до чужих даних або ресурсів.

3. Платформений рівень (Platform Layer) - включає операційні системи, бази даних, API та сервіси, які використовуються для розроблення застосунків. Тут важливою є безпека середовища виконання, контроль оновлень, патчів та правильна ізоляція між модулями.
4. Прикладний рівень (Application Layer) - забезпечує доступ кінцевих користувачів до застосунків і сервісів. На цьому рівні основна увага приділяється контролю доступів (IAM), шифруванню трафіку, веденню журналу подій та політикам доступу до даних.

Хмарні провайдери реалізують захист на кожному з цих рівнів відповідно до принципів «security by design» і «defense in depth», однак користувач є відповідальним за правильне налаштування, використання та моніторинг ресурсів. Цей принцип закріплений у концепції моделі спільної відповідальності (Shared Responsibility Model).

Модель спільної відповідальності (Shared Responsibility Model)

Модель спільної відповідальності визначає межу між зонами відповідальності постачальника хмарних послуг та користувача. Її основна мета - запобігти непорозумінням щодо того, хто саме відповідає за конкретні аспекти безпеки.

У загальному вигляді:

- Провайдер відповідає за безпеку саме хмарних ресурсів - тобто за фізичну безпеку дата-центрів, інфраструктуру, віртуалізаційний шар, гіпервізори, мережеві з’єднання та системи зберігання.
- Користувач відповідає за безпеку в хмарному середовищі - тобто за налаштування облікових записів, контроль доступу (IAM), шифрування даних, політики мережевої безпеки, моніторинг подій і дотримання нормативних вимог.

Розподіл обов’язків залежить від моделі сервісу:

- У IaaS користувач керує операційними системами, мережевими політиками, сховищами даних та налаштуваннями безпеки; провайдер відповідає лише за інфраструктуру.
- У PaaS користувач відповідає за код, контейнери, конфігурації застосунків і керування обліковими записами; постачальник - за саму платформу, оновлення й ізоляцію середовища.
- У SaaS провайдер бере на себе майже всю відповідальність за безпеку, а користувач лише контролює доступи, паролі, двофакторну автентифікацію та політики зберігання даних.

Вплив моделі відповідальності на безпеку

Порушення принципів моделі спільної відповідальності є однією з найпоширеніших причин інцидентів безпеки у хмарних середовищах.
Наприклад, витоки даних у сервісах Amazon S3 майже завжди спричинені неправильно налаштованими політиками доступу, а не технічними збоями з боку AWS. Аналогічно, відсутність шифрування баз даних або журналу подій зазвичай є наслідком помилок користувача.

Отже, ефективність захисту хмарного середовища залежить не лише від можливостей провайдера, а й від зрілості процесів безпеки в організації користувача. Ключовим завданням сучасних систем управління безпекою є автоматизація перевірки налаштувань і забезпечення постійної відповідності моделі спільної відповідальності, що надалі буде розглянуто в контексті розробки рекомендацій і програмного засобу для аналізу конфігурацій AWS.

1.1.3. Основні вимоги до захисту інформації в хмарному середовищі

Захист інформації в хмарних обчисленнях має на меті забезпечення цілісності, конфіденційності, доступності та підзвітності даних, що зберігаються або обробляються у середовищах сторонніх провайдерів. Ці вимоги закріплені у міжнародних стандартах, зокрема ISO/IEC 27001, ISO/IEC 27017, NIST SP 800-53 Rev.5 та CIS Benchmarks, і становлять основу для побудови політик безпеки в хмарі.

1. Конфіденційність (Confidentiality)

Конфіденційність передбачає обмеження доступу до інформації лише для уповноважених користувачів і процесів. У хмарних середовищах це досягається завдяки:

- використанню механізмів керування ідентичностями та доступом (Identity and Access Management, IAM);
- застосуванню багатофакторної автентифікації (MFA) та принципу least privilege (мінімуму необхідних прав);
- шифруванню даних як у стані спокою (at rest), так і під час передавання (in transit);
- контролю політик доступу до об’єктів і сервісів (наприклад, блокування публічних S3-бакетів).

Порушення конфіденційності найчастіше відбувається через людський фактор - неправильні налаштування політик доступу або випадкову публікацію секретів у відкритих репозиторіях.

2. Цілісність (Integrity)

Цілісність означає збереження незмінності, повноти та достовірності даних протягом усього життєвого циклу.
У хмарних системах вона забезпечується:

- контролем цілісності даних на рівні зберігання та передачі (контрольні суми та цифрові підписи);

- механізмами версіонування та журналювання операцій (наприклад AWS CloudTrail, Config);
- застосуванням KMS-ключів із ротацією для захисту даних від несанкціонованих змін;
- аудитом доступу до об’єктів і ресурсів.

Порушення цілісності може бути наслідком як навмисних дій (атаки типу “data tampering”), так і технічних збоїв без резервування чи перевірки контрольних хешів.

3. Доступність (Availability)

Доступність - здатність системи надавати ресурси та сервіси у потрібний час, навіть у разі часткових збоїв чи атак. У контексті хмарних середовищ вона досягається завдяки:

- географічному резервуванню ресурсів у різних регіонах та зонах доступності (Availability Zones);
- автоматичному масштабуванню й балансуванню ресурсів в залежності від навантаження;
- системам резервного копіювання та відновлення;
- моніторингу продуктивності й вчасному реагуванню на інциденти.

Атаки типу DDoS чи помилки в конфігураціях мережевих політик можуть безпосередньо впливати на доступність критичних сервісів.

4. Підзвітність та аудит (Accountability)

Підзвітність передбачає можливість простежити всі дії користувачів і системних процесів, що мають вплив на безпеку. Для цього у хмарних середовищах впроваджуються:

- централізоване журналювання та аудит подій (AWS CloudTrail, Azure Monitor, Google Cloud Audit Logs);
- моніторинг аномалій і виявлення загроз (AWS GuardDuty, Security Command Center тощо);
- механізми сповіщень та реагування (AWS SNS, EventBridge);
- звітність щодо відповідності вимогам стандартів (compliance reports в AWS Security Hub).

Ця вимога особливо важлива для організацій, які підпадають під регуляції GDPR, ISO 27001 чи PCI DSS.

5. Нормативно-правова відповідність (Compliance)

Організації, що працюють в хмарних середовищах, повинні дотримуватись вимог локального законодавства та міжнародних стандартів [4]. Провайдери, як-от AWS, Azure, GCP, надають інструменти для контролю відповідності (Compliance Center, Artifact, Security Hub). Відповідальність за дотримання

політик приватності, захисту персональних даних і термінів зберігання інформації лежить на організації і не контролюється провайдером.

Узагальнення

Таким чином, ефективна система захисту інформації в хмарному середовищі повинна забезпечувати комплексне виконання всіх вимог CIA-тріади (Confidentiality, Integrity, Availability), а також підзвітність та відповідність нормативам. Реалізація цих вимог вимагає поєднання технічних засобів, політик доступу, моніторингу та регулярного аудиту, що створює підгрунтя для подальшого аналізу принципів побудови безпечних хмарних систем у наступних розділах роботи.

1.1.4. Принципи забезпечення конфіденційності, цілісності та доступності даних у хмарних системах

Хмарне середовище передбачає спільне використання обчислювальних ресурсів багатьма користувачами, тому питання захисту даних набуває особливого значення. Для підтримання стабільності, довіри та безпеки хмарних сервісів використовують базові принципи, які формують основу будь-якої системи захисту - конфіденційність, цілісність і доступність (так звана CIA-тріада). Ці принципи не існують ізольовано, а взаємодіють між собою, утворюючи цілісну модель управління ризиками в хмарній інфраструктурі.

Принцип конфіденційності

Конфіденційність означає, що доступ до інформації мають лише ті особи або процеси, яким він був наданий. У хмарних системах цей принцип реалізується через поєднання технічних і організаційних заходів: керування обліковими записами та ролями користувачів, багатофакторну автентифікацію. Серед технічних заходів використовують сегментацію мереж, шифрування даних(в спокою і при передачі) і постійний моніторинг активності.

Особливу увагу приділяють політикам доступу - навіть незначна помилка в налаштуваннях може призвести до витоку конфіденційних даних, що вже неодноразово ставало причиною інцидентів у сервісах типу Amazon S3 чи Google Cloud Storage.

Принцип цілісності

Цілісність полягає у гарантії того, що інформація залишається точною, повною і незмінною з моменту створення до моменту використання.У хмарних середовищах цей принцип забезпечується контролем версій даних, перевіркою контрольних сум, застосуванням цифрових підписів і аудитом усіх змін.

Важливим механізмом є журналювання операцій - наприклад, служби AWS CloudTrail або Azure Monitor дозволяють відстежувати, хто і коли змінював налаштування або конфігурації. Дотримання принципу цілісності допомагає запобігти як навмисним маніпуляціям даними, так і випадковим помилкам, що можуть порушити роботу системи.

Принцип доступності

Доступність означає, що дані та сервіси повинні бути доступні користувачам у потрібний момент, навіть у випадку часткових збоїв або кібератак. Хмарні провайдери забезпечують цей принцип через географічне резервування, автоматичне масштабування ресурсів і системи відновлення після збоїв. Водночас відповідальність користувача полягає у правильному налаштуванні політик відмовостійкості, резервного копіювання та контролю навантаження. Також користувачам дуже важливо правильно налаштовувати горизонтальне і вертикальне масштабування та розгортати додаток в декількох зонах доступності, це допоможе уникнути проблем з недостачою ресурсів. Недотримання принципу доступності може призвести до простоїв критичних сервісів або втрати довіри клієнтів.

Баланс між принципами

Забезпечення цих трьох принципів у хмарному середовищі потребує балансу. Підвищення конфіденційності через надмірні обмеження може знизити доступність, тоді як пріоритет швидкодії - ослабити контроль за доступами. Завдання фахівця з безпеки в хмарних середовищах полягає в тому, щоб знайти оптимальне співвідношення між конфіденційністю, цілісністю та доступністю з урахуванням бізнес-цілей організації, типу даних і обраної моделі хмарного розгортання.

1.1.5. Роль політик безпеки та контролів на рівні сервісів і користувачів.

Безпека хмарного середовища неможлива без чітко визначеної системи політик безпеки - набору правил, вимог і процедур, які регулюють порядок роботи з інформацією, доступ до ресурсів і способи реагування на інциденти. Саме політики створюють зв’язок між організаційними процесами, технічними контролями й реальними діями користувачів у хмарі.

Політики безпеки в хмарних системах виконують кілька ключових функцій:

- визначають межі відповідальності між провайдером і користувачем згідно з моделлю спільної відповідальності;
- встановлюють правила доступу до даних та ресурсів відповідно до ролей і рівнів доступу;
- формують вимоги до аудиту, моніторингу, журналювання та зберігання подій безпеки;
- задають стандарти шифрування, резервування, оновлень і реагування на інциденти.

Відповідно до сучасних підходів, політики безпеки реалізуються через систему контролів - конкретних технічних або адміністративних механізмів, які виконують ці вимоги на практиці. У хмарних середовищах вони зазвичай поділяються на такі групи:

1. Адміністративні правила - організаційні правила, інструкції та процедури, що регулюють управління безпекою. Наприклад, політики створення облікових записів, управління доступом, порядок реагування на інциденти чи перевірка персоналу.
2. Технічні правила - механізми, реалізовані безпосередньо у хмарній інфраструктурі:
   - керування ідентичностями та доступом (IAM);
   - шифрування даних (KMS, TLS, SSE);
   - міжмережеві політики безпеки (Security Groups, Network ACLs, Firewall Rules);
   - моніторинг і виявлення загроз (CloudTrail, GuardDuty, Security Hub);
   - автоматичне виявлення місконфігурацій і контроль відповідності (Config, Conformance Packs).
3. Операційні правила - заходи, що підтримують безпечне функціонування систем у повсякденній роботі. Приклади: регулярні оновлення, перевірка журналів, відновлення після збоїв, тестування планів реагування на інциденти.

У сучасних хмарних архітектурах політики безпеки дедалі частіше реалізуються через автоматизацію, тобто у вигляді коду (Security as a Code). Це дає змогу інтегрувати контролі безпеки безпосередньо у процеси розгортання інфраструктури (IaC) і DevOps-пайплайни. Наприклад, політики доступу або вимоги до шифрування можуть бути описані в YAML-конфігураціях і автоматично перевірятися під час кожного деплою.

Важливо, що навіть найсучасніші технічні засоби не гарантують безпеки. Якщо відсутня політика на рівні користувачів і відсутня культура безпеки всередині організації, то ризики стають більш ймовірними. Людський фактор - одна з головних причин інцидентів у хмарі: недбале ставлення до паролів, порушення процедур доступу, нехтування багатофакторною автентифікацією чи невчасне оновлення налаштувань. Тому поряд із технічними контролями необхідно забезпечувати постійне навчання персоналу, внутрішній аудит і культуру безпеки всередині організації.

Отже, політики безпеки та пов’язані з ними контролі є центральним елементом системи управління ризиками в хмарних середовищах. Вони забезпечують єдність підходів до захисту, дозволяють зменшити вплив людських помилок і створюють основу для подальшої автоматизованої перевірки конфігурацій, яку розроблено у межах даного дослідження.

1.2. Сучасні загрози та уразливості хмарних середовищ

1.2.1. Класифікація загроз у хмарних середовищах

Загрози інформаційній безпеці в хмарі можна поділити за кількома ознаками: за джерелом походження, за рівнем впливу, за технічними особливостями реалізації та за моделлю сервісу (IaaS, PaaS, SaaS).

1. За джерелом виникнення
- Зовнішні загрози - атаки з боку хакерів, конкурентів чи автоматизованих ботнетів, спрямовані на порушення конфіденційності або доступності даних. Сюди належать спроби несанкціонованого доступу до акаунтів, фішинг, DDoS-атаки, експлуатація відкритих портів чи API-інтерфейсів.
- Внутрішні загрози (інсайдерські) - дії співробітників або підрядників, які мають законний доступ до хмарної інфраструктури, але використовують його зловмисно або необережно. Прикладом є випадкове видалення даних чи навмисне розголошення конфіденційної інформації.
- Технологічні загрози - пов’язані з помилками у віртуалізації, некоректною ізоляцією контейнерів, вразливостями у бібліотеках або сторонніх залежностях.
- Організаційні загрози - виникають через недотримання політик безпеки, відсутність контролю змін або недостатню кваліфікацію персоналу.

2. За рівнем впливу
- Загрози конфіденційності - витік персональних даних, розкриття комерційної інформації, компрометація облікових даних.
- Загрози цілісності - несанкціоновані зміни у конфігураціях або даних, маніпуляції журналами подій, порушення коректності транзакцій.
- Загрози доступності - відмова сервісів, перевантаження ресурсів, пошкодження сховищ або атаки на мережеву інфраструктуру.

3. За моделлю надання послуг
- У моделі IaaS найбільшу небезпеку становлять неправильні мережеві налаштування, відкриті порти, слабкі паролі адміністративних облікових записів і неоновлені віртуальні машини.
- У PaaS типові ризики пов’язані з некоректними дозволами в середовищах виконання, вразливими бібліотеками та помилками у застосунковій логіці.
- У SaaS основний фокус - на управлінні користувачами та запобіганні несанкціонованому доступу до облікових записів, особливо під час інтеграції з іншими системами через API.

1.2.2. Типові вразливості хмарних середовищ

Згідно з результатами міжнародних досліджень (CSA Top Threats 2024, ENISA 2023, Hussain et al., 2021), найбільш поширеними є такі категорії вразливостей:

1. Помилки конфігурації (Misconfigurations)
Відкриті S3-бокети, неправильні політики IAM, необмежені мережеві правила (0.0.0.0/0) - усе це залишається найчастішою причиною інцидентів.
2. Надлишкові привілеї та слабкий контроль ідентичностей
Використання ролей з правами Action: "*", відсутність багатофакторної автентифікації, тривале використання статичних Access Keys.
3. Відсутність належного моніторингу та журналювання
Вимкнений CloudTrail або відсутність збору логів робить неможливим виявлення несанкціонованих дій.
4. Недостатній рівень шифрування
Незашифровані сховища, бази даних або знімки томів (snapshots) можуть бути доступними для сторонніх осіб.
5. Компрометація облікових даних і секретів
Зберігання ключів доступу у вихідному коді, відкритих репозиторіях або змінних середовища без обмежень доступу.
6. Ланцюг постачання (Supply Chain Risks)
Використання сторонніх бібліотек чи контейнерних образів із вразливими компонентами.
7. Інсайдерські ризики
Недбалі або навмисні дії співробітників, що мають надмірні права доступу.

Кожен із цих чинників може виступати початковою точкою компрометації системи та потребує постійного контролю.

У наступних підрозділах цього розділу буде детальніше розглянуто окремі категорії вразливостей, їхні наслідки та сучасні підходи до їхнього виявлення і запобігання, включно з порівнянням рекомендацій провідних фреймворків - CIS Benchmarks, NIST SP 800-53 та AWS Foundational Security Best Practices.

1.2.3 Основні типи уразливостей та їхній вплив на безпеку хмарної інфраструктури

Попри високий рівень технологічного захисту з боку провайдерів, більшість інцидентів у хмарних середовищах виникають через помилки користувачів або недоліки конфігурацій. Хмара дає значну свободу у налаштуваннях, але саме ця гнучкість часто стає джерелом ризику. Нижче наведено найпоширеніші типи уразливостей, які фіксуються у практиці експлуатації хмарних сервісів, та їхній вплив на загальний стан безпеки.
1. Помилки конфігурації (Misconfigurations)
Неправильне налаштування політик доступу або ресурсів залишається однією з найчастіших причин витоку даних.

Типовими прикладами є публічні S3-бакети з незашифрованими файлами, відкриті порти у Security Groups, надто широкі дозволи для користувачів чи сервісних ролей.

Такі помилки зазвичай не пов’язані з технологічними вадами самої хмари, а виникають через неуважність або відсутність контролю змін.

Вплив:
- несанкціонований доступ до конфіденційної інформації;
- можливість модифікації або видалення критичних даних;
- компрометація корпоративних облікових записів і сервісів.

За даними CSA (2024), саме неправильні конфігурації становлять понад 40% інцидентів безпеки у публічних хмарах. Наприклад, у 2023 році дослідники виявили понад 20 000 публічно доступних S3-бакетів із відкритими клієнтськими даними - від фінансових документів до сканів паспортів.

2. Надлишкові привілеї та вразливості в системах керування ідентичностями (IAM)

Механізми керування доступом є серцем будь-якої хмарної інфраструктури. Однак на практиці користувачі часто отримують ширші права, ніж потрібно для їхніх завдань.

Найбільш небезпечними є політики з параметрами "Action": "*" або "Resource": "*", які фактично надають повний адміністративний доступ.
Інша поширена проблема - використання довготривалих статичних ключів доступу без ротації або вимкненої багатофакторної автентифікації (MFA).

Вплив:
- повний контроль зловмисника над ресурсами в разі компрометації облікових даних;
- можливість створення нових користувачів чи ролей;
- приховані дії в межах легітимного доступу, які складно виявити.

Особливо небезпечними є сценарії, коли зловмисник отримує токен із надлишковими правами через неправильно налаштований сервіс або IAM роль, наприклад, у контейнерному середовищі без ізоляції метаданих.

3. Відсутність журналювання та моніторингу

Без централізованого збору журналів і подій безпеки організація фактично “сліпа” до внутрішніх змін у хмарі.

Сервіси на кшталт AWS CloudTrail, Azure Activity Logs або Google Cloud Audit Logs повинні бути активовані у всіх регіонах і збирати повний набір подій доступу.

Нерідко виявляється, що ці сервіси вимкнені або налаштовані лише частково, що унеможливлює подальше розслідування інцидентів.

Вплив:
- відсутність доказової бази у разі витоку чи компрометації;
- неможливість виявити аномальні дії користувачів;

- затримка у реагуванні на інциденти.

Прикладом є ситуація, коли атака на адміністративну консоль залишилася непоміченою протягом декількох днів, оскільки CloudTrail не збирав події у регіоні, де зловмисник виконував запити.

4. Недостатній рівень шифрування та управління ключами

Шифрування даних - одна з базових вимог, але на практиці часто залишається необов’язковим або неправильно реалізованим.
Типовими помилками є відсутність default encryption для S3, EBS або RDS, зберігання відкритих копій ключів чи нерегулярна ротація KMS-ключів.

Вплив:
- витік незашифрованих даних у разі компрометації сховища;
- підвищений ризик несанкціонованого відновлення резервних копій;
- невідповідність стандартам безпеки (наприклад, GDPR або ISO 27018).

У кількох відомих інцидентах дані компаній опинилися у відкритому доступі саме через відсутність шифрування резервних знімків або логів.

5. Компрометація облікових даних і секретів

Однією з найпоширеніших помилок розробників є зберігання секретів (паролів, токенів, API-ключів) у відкритих репозиторіях коду або у змінних середовища без шифрування.
Іноді такі ключі випадково публікуються на GitHub чи Docker Hub, після чого зловмисники автоматично їх виявляють і використовують.

Вплив:
- повна компрометація акаунтів або контейнерних середовищ;
- несанкціонований запуск віртуальних машин чи сервісів за рахунок жертви;
- фінансові втрати внаслідок використання ресурсів (наприклад, для майнінгу).

Ці інциденти особливо небезпечні в середовищах DevOps, де автоматизація деплою здійснюється за допомогою токенів з розширеними правами.

6. Вразливості ланцюга постачання (Supply Chain Risks)

Хмарні сервіси активно інтегруються із зовнішніми бібліотеками, контейнерними образами та сервісами сторонніх постачальників.
Якщо один із компонентів містить вразливість, вона може поширитися на всю систему.

Показовим прикладом є випадки зараження контейнерних образів у публічних реєстрах або компрометація бібліотек у відкритих репозиторіях.

Вплив:

- поширення шкідливого коду через легітимні процеси CI/CD;
- складність виявлення джерела компрометації;
- потенційна атака на довірчі відносини між сервісами.

Через це великі провайдери запроваджують програми перевірки постачальників і валідації контейнерів (наприклад, AWS Verified Publisher, Docker Content Trust).

7. Інсайдерські загрози

Інсайдерські інциденти трапляються рідше, але їхній вплив може бути найсерйознішим.
Йдеться не лише про навмисні дії - наприклад, злив даних співробітником, - а й про звичайну недбалість: випадкове видалення ресурсів, невдале оновлення політик або поширення паролів між членами команди.

Вплив:

- витік критичної інформації;
- знищення або спотворення даних;
- порушення безперервності бізнес-процесів.

Для мінімізації таких ризиків застосовують принцип separation of duties (розподіл обов’язків), обмеження прав адміністраторів та обов’язковий аудит усіх адміністративних дій.

Узагальнення

Більшість уразливостей у хмарних середовищах виникають не через відсутність засобів захисту, а через їхнє неправильне використання або ігнорування.
Практика показує, що навіть організації, які впроваджують політики безпеки, часто не мають автоматизованих механізмів перевірки дотримання цих політик.
Саме тому сучасна тенденція у сфері кібербезпеки спрямована на впровадження Cloud Security Posture Management (CSPM) - рішень, що здійснюють постійний аудит конфігурацій та попереджають адміністратора про відхилення від безпечних налаштувань.

У наступному розділі буде розглянуто, як існуючі фреймворки безпеки (CIS, NIST, AWS FSBP) охоплюють зазначені проблеми та які обмеження вони мають у контексті автоматизації контролю хмарних конфігурацій.

1.3. Аналіз існуючих методів та стандартів забезпечення безпеки

1.3.1. Огляд фреймворків і їх роль

Екосистема хмарної безпеки спирається на кілька взаємодоповнювальних рамок. CIS Benchmarks задають “пороговий” рівень гігієни конфігурацій, AWS Foundational Security Best Practices (FSBP) надає автоматизовані перевірки для сервісів AWS, NIST SP 800-53 описує універсальний каталог контролів для проєктування систем захисту, ISO/IEC 27017/27018 фіксують організаційні вимоги для провайдерів і споживачів хмарних послуг, а CSA Cloud Controls Matrix (CCM) узгоджує терміни та мапує вимоги між різними стандартами. Разом вони формують базу для політик, аудитів та технічних контрольних списків.

CIS Amazon Web Services Foundations Benchmark

Призначення. Практичний набір контрольних перевірок для акаунтів і ключових сервісів AWS (ідентичності, журналювання, мережа, сховище).
Сильні сторони:
- Конкретні, перевіряльні вимоги (on/off, enabled/disabled), які легко автоматизувати.
- Добре підходить як стартовий “базовий рівень” для нових акаунтів або аудитів.
- Широко підтримується комерційними та open-source інструментами (сканери, CSPM).

Обмеження:
- Орієнтація на мінімальний рівень захисту; не враховує контекст ризиків організації.
- Частина контролів статичні: не завжди відображають особливості робочих навантажень і винятки.
- Потребує доповнення політиками на рівні процесів (IAM-процедури, реагування, тренінги).

AWS Foundational Security Best Practices (FSBP) / Security Hub

Призначення. Набір вбудованих у AWS Security Hub автоматизованих перевірок із “security score”, що агрегує події з GuardDuty, Config, Inspector та ін.
Сильні сторони:
- Автоматизація “із коробки”: регулярні тести, центральна панель, пріоритизація findings.
- Глибша інтеграція з AWS-сервісами, ніж у загальних бенчмарків; оперативні рекомендації.
- Підтримка мапінгів на NIST/PCI/ISO профілі для звітності.
- Обмеження:
- Прив’язка до AWS: обмежена переносимість для мультихмарних середовищ.

– Не всі організаційні вимоги покриваються технічними чеками; потрібні додаткові процеси.
– За замовчуванням фокусується на наявності/стані налаштувань, а не на бізнес-ризику.

NIST SP 800-53

Призначення. Універсальний каталог контролів (управлінських, операційних, технічних) для систем і організацій, незалежно від хмари чи он-прем.
Сильні сторони:
- Повнота й системність: охоплює політики, персонал, процедури, технології.
- Добре підходить для проєктування системи захисту та доказової відповідності.

Обмеження:
- Низька “з коробки” автоматизованість: потребує мапінгу на конкретні сервіси/перевірки.
- Високі витрати на впровадження та підтримку без спеціалізованих інструментів.
- Абстрактність формулювань: без локальної інтерпретації важко перевести у чек-листи.

ISO/IEC 27017 та ISO/IEC 27018

Призначення. Додаткові вимоги до ISMS у хмарі (27017) та захист персональних даних у публічних хмарах (27018).
Сильні сторони:
- Зрозумілий організаційний каркас: ролі, договори, обов’язки провайдера/замовника.
- Корисно для підготовки до сертифікацій і взаємодії з аудиторами/замовниками.

Обмеження:
- Менше технічних деталей; потрібна технічна специфікація/чек-листи на рівні платформи.
- Не забезпечує автоматичної оцінки постури - це радше рамка управління.

CSA Cloud Controls Matrix (CCM)

Призначення. “Словник відповідностей” між вимогами різних стандартів із фокусом на хмару; допомагає з уніфікацією термінів і трасуванням контролів.
Сильні сторони:
– Прискорює мапінг: від бізнес-вимог до технічних перевірок.
– Підтримує мультистандартне середовище (NIST/ISO/PCI/ENISA тощо).

Обмеження:

- Не є інструментом перевірки; це довідкова матриця, яку треба реалізувати технічно.
- Якість мапінгу залежить від зрілості локальних політик і інструментів.

Порівняння за ключовими критеріями

- Автоматизація: FSBP (висока, вбудована в AWS) > CIS (середня, залежить від інструмента) > ISO/NIST/CCM (низька без мапінгів).
- Глибина технічних перевірок: FSBP ≈ CIS (сервіс-специфічні, перевіряльні) > NIST/ISO/CCM (процесні).
- Переносимість (multi-cloud): NIST/ISO/CCM (висока) > CIS (провайдер-специфічні профілі) > FSBP (AWS-центричний).
- Зручність аудиту/репортингу: NIST/ISO (сильні в доказовій базі), FSBP/CIS - зручні як технічний додаток до аудиту.
- Вартість впровадження: CIS/FSBP (швидкий старт) < ISO/NIST (проектний підхід, навчання, інтеграції).

1.3.2. Висновки для практичного застосування

1. Комбінований підхід дає найкращий ефект. NIST/ISO задають “дах” процесів і відповідальностей; CIS/FSBP - щоденну перевірку технічних налаштувань; CSA CCM - клей між ними.
2. Безперервність важливіша за разові аудити. Автоматизовані скани (FSBP/CIS-aligned правила) потрібно запускати постійно, з фіксацією відхилень і трендами.
3. Контекст і пріоритизація. Сам факт “червоного” чеку не дорівнює високому ризику. Потрібні ваги/пріоритети за впливом на бізнес-активи й типи даних.
4. Мультихмара вимагає власного шару абстракції. Для GCP/Azure доведеться віддзеркалити набір CIS-подібних перевірок і звести їх під єдину модель findings та мапінгів.
5. Інтеграція з DevSecOps. Частину контролів варто перетворити на перевірки “перед деплоєм” (policy-as-code), щоб ловити помилки до потрапляння у прод.

Таблиця 1. Взаємозв’язок загроз, контролів і технічних перевірок у хмарному середовищі

№	Типова загроза / уразливість	Відповідні контролі (NIST SP 800-53, ISO/IEC 27017, CSA CCM)	Приклади технічних перевірок (CIS Benchmark / AWS FSBP)	Очікуваний ефект від реалізації
1	Помилки конфігурації ресурсів (misconfigurations)	NIST CM-2 “Baseline Configuration”; ISO 27017 A.9.5; CCM SEF-03	CIS 1.4 - Ensure S3 Block Public Access is enabled; FSBP S3.1 - S3 buckets should prohibit public access	Запобігання несанкціонованому доступу до відкритих сховищ і витоку даних
2	Надлишкові привілеї, слабке IAM-керування	NIST AC-2, AC-6; ISO 27017 A.9.2; CCM IAM-01	CIS 1.14 - MFA enabled for root account; FSBP IAM.5 - Avoid wildcards in policies	Мінімізація ризику компрометації облікових записів, зменшення поверхні атаки
3	Відсутність журналювання та моніторингу	NIST AU-2, AU-6; ISO 27017 A.12.4; CCM LOG-01	CIS 2.1 - CloudTrail enabled in all regions; FSBP CT.1 - CloudTrail should be enabled and encrypted	Забезпечення підзвітності, виявлення несанкціонованих дій
4	Недостатній рівень шифрування даних	NIST SC-12, SC-13; ISO 27018 A.10.1; CCM EKM-02	CIS 2.6 - Ensure S3 default encryption enabled; FSBP S3.7 - S3 buckets should use SSE or KMS	Гарантована конфіденційність даних “at rest” і “in transit”, зменшення наслідків витоків

5	Компрометація облікових даних / секретів	NIST IA-5; ISO 27017 A.9.3; CCM KMS-01	FSBP IAM.8 - Rotate access keys every 90 days; CIS 1.9 - Avoid hard-coded credentials	Зменшення ризику використання викрадених токенів або старих ключів
6	Вразливості ланцюга постачання (supply chain)	NIST SR-3, SA-12; ISO 27017 A.15.1; CCM DSP-05	FSBP CodeBuild.1 - Build projects should use encryption and signed sources	Контроль походження компонентів, запобігання впровадженню шкідливого коду
7	Інсайдерські загрози та людський фактор	NIST PS-6, AT-2; ISO 27017 A.7.2; CCM HRS-03	CIS 1.12 - Ensure no unused credentials exist; FSBP IAM.6 - Deactivate stale users	Підвищення прозорості дій користувачів, зниження ризику зловживань
8	Порушення доступності сервісів (DoS, збої)	NIST CP-2, CP-10; ISO 27017 A.17.1; CCM BCR-02	FSBP EC2.8 - Ensure Auto Recovery enabled; CIS 3.1 - Multi-AZ deployments for critical resources	Підвищення відмовостійкості та швидке відновлення після інцидентів

1.3.3. Статистичні дані щодо інцидентів у хмарних середовищах

Оцінка масштабів інцидентів у хмарних середовищах на основі авторитетних джерел дає змогу краще усвідомити тренди та вразливості, які потребують уваги. Нижче наведено ключові статистичні дані та висновки за останні роки від Cloud Security Alliance (CSA), IBM X-Force та European Union Agency for Cybersecurity (ENISA).

У звіті «Top Threats to Cloud Computing 2024» CSA зазначає, що за результатами опитування понад 500 фахівців із безпеки окремо виділили топ-11 загроз для хмарних обчислень. Найбільшою небезпекою визнані misconfigurations та недостатній контроль змін (Misconfiguration and inadequate

change control), які очолили рейтинг загроз. IAM (Identity & Access Management) посіла друге місце, а нестабільні інтерфейси та API - третє. Ці дані свідчать про те, що ключові інциденти у хмарі пов’язані не з внутрішніми технічними вразливостями провайдерів, а з налаштуваннями та політиками користувачів/організацій.

У звіті IBM X-Force Cloud Threat Landscape 2023 IBM виділяє, що «невірне використання облікових даних» (improper use of credentials) стало топ-причиною компрометацій у хмарних середовищах за минулий рік. Також відзначається, що хоча атаки на основі штучного інтелекту у хмарі наразі утримуються на помірному рівні, існують передумови до їхнього зростання. Центральною точкою стає те, що навіть “прості” методи - викрадення ключів доступу, фішинг, повторне використання сервісних ролей - залишаються високо ефективними.

У звіті ENISA Threat Landscape 2023 агентства ENISA («11-те видання») аналізовано тисячі інцидентів кібербезпеки у Європейському Союзі. Зокрема, зазначено, що хмарні середовища дедалі частіше виступають як вектор атак - наприклад, спеціалізовані віртуальні машини використовуються для ботнетів або DDoS-операцій. Також ENISA підкреслює, що недостатній контроль над хмарною інфраструктурою провокує ланцюгові ефекти - від мережевих перебоїв до витоків персональних даних.

Висновок із статистики

- Помічається чітка тенденція: класичні помилки конфігурацій і доступів (misconfigurations + IAM) продовжують залишатися лідируючими чинниками інцидентів.
- Автоматизація, видимість і контроль змін у хмарі - критичні компоненти, які здатні знизити частоту випадків компрометації.
- Незважаючи на появу нових атак (AI-інструменти, ланцюги постачання, ботнети в хмарі), наразі найбільшою загрозою є людський фактор і неправильні налаштування.
- Організації, які володіють показниками і логами (IAM, журналювання, зміни в конфігураціях) мають суттєву перевагу у ранньому виявленні та реагуванні.

1.3.4. Приклади реальних інцидентів у великих організаціях

Статистичні дані, наведені у звітах Cloud Security Alliance, IBM X-Force та ENISA, підтверджуються низкою публічних інцидентів, що сталися у відомих міжнародних компаніях. Аналіз таких випадків дозволяє простежити спільні закономірності та показує, що більшість порушень у хмарному середовищі пов’язані з людським фактором, помилками конфігурацій і неналежним контролем доступу.

Одним із найвідоміших прикладів є інцидент у банку Capital One, що стався у 2019 році. Унаслідок витоку були скомпрометовані персональні дані

понад ста мільйонів клієнтів зі США та Канади. Розслідування показало, що зловмисниця, колишня працівниця компанії Amazon Web Services, скористалася вразливістю типу Server-Side Request Forgery у веб-додатку Capital One, отримавши доступ до метаданих і облікових токенів IAM-ролей. Надлишкові права, надані ролям у хмарному акаунті, дозволили їй завантажити великі обсяги конфіденційних даних із S3-сховищ. Цей інцидент став класичним прикладом поєднання технічної вразливості з людською помилкою в управлінні політиками доступу.

Схожі наслідки мало порушення безпеки в готельній мережі Marriott International, про яке було оголошено у 2018 році. Після придбання компанії Starwood Hotels & Resorts, Marriott успадкувала її хмарну IT-інфраструктуру, де вже тривалий час існувала вразливість у системі керування обліковими записами. Хакери непомітно діяли у цій мережі понад чотири роки, отримуючи доступ до персональних даних клієнтів. Основною причиною інциденту стала відсутність централізованого моніторингу та слабкий контроль змін після інтеграції двох систем. Цей випадок підкреслив, наскільки важливо під час злиттів і придбань проводити повний аудит безпеки хмарних ресурсів.

Ще один показовий приклад - інцидент у пенсійному фонді UniSuper (Австралія), який стався у 2024 році внаслідок помилки конфігурації в Google Cloud. Через неправильне налаштування приватного хмарного акаунта відбулося повне видалення середовища організації, унаслідок чого понад 600 тисяч користувачів тимчасово втратили доступ до сервісів. Хоча дані не були скомпрометовані, подія показала, що навіть технічна помилка без злому може мати значний вплив на доступність (availability) та репутацію компанії. Інцидент також підтвердив: автоматизація не знімає потреби в контролі змін і резервуванні даних.

У 2021 році виявлено ще один масовий інцидент, який зачепив низку великих організацій, серед яких American Airlines, Ford, New York City Department of Education та інші. Проблема полягала у відкритих конфігураціях сервісу Microsoft Power Apps, де за замовчуванням API інтерфейси були доступні без автентифікації. Унаслідок цього понад 38 мільйонів записів (контактні дані, ідентифікатори, адреси електронної пошти) опинилися у вільному доступі. Причиною стала типова помилка налаштувань (misconfiguration) у середовищі SaaS, коли користувачі не перевірили політики публічного доступу до елементів додатків. Цей кейс показав, що навіть у середовищах високого рівня абстракції (PaaS / SaaS) залишається потреба у відповідальному управлінні політиками безпеки.

Не менш важливим прикладом є випадок із SingHealth - найбільшою державною медичною організацією Сінгапуру. У 2018 році стався витік медичних даних 1,5 мільйона пацієнтів, включно з персональними даними прем’єр-міністра країни. Розслідування показало, що хакери скористалися

слабким контролем доступу на внутрішньому сервері та відсутністю багатофакторної автентифікації. Крім того, системи моніторингу були налаштовані фрагментарно, що дозволило зловмисникам діяти тривалий час непомітно. Цей інцидент підтвердив: відсутність журналювання і прозорості доступів призводить до масштабних наслідків навіть без прямої технічної вразливості.

Усі наведені випадки демонструють, що незалежно від розміру організації чи хмарного провайдера, головними чинниками інцидентів залишаються помилки конфігурацій, надлишкові права доступу та відсутність належного моніторингу. Ці тенденції повністю збігаються зі статистичними спостереженнями CSA, IBM та ENISA і підкреслюють необхідність системного підходу до перевірки налаштувань безпеки, автоматизованого аудиту та навчання персоналу.

1.3.5. Виявлення проблем і недоліків у сучасних підходах

Проведений аналіз літератури, статистичних звітів та існуючих стандартів безпеки показав, що, незважаючи на значний прогрес у розвитку інструментів і методик, сучасна практика захисту хмарних середовищ має низку суттєвих обмежень. Ці недоліки проявляються як у технічній, так і в організаційній площині, що свідчить про потребу в нових підходах до інтегрованого управління ризиками.

Однією з найбільш очевидних проблем є фрагментарність існуючих засобів контролю. Переважна більшість інструментів орієнтована на перевірку конфігурацій окремих сервісів або хмарних провайдерів. Наприклад, AWS Security Hub чи Azure Security Center надають детальні перевірки для власної платформи, проте не забезпечують уніфікованої моделі для мультихмарних середовищ. У результаті організації, що використовують одночасно кілька провайдерів, стикаються з розрізненими звітами, дублюванням даних і відсутністю єдиної картини ризиків. Це ускладнює централізоване управління безпекою і знижує ефективність реагування на інциденти.

Другою суттєвою проблемою є обмежена інтеграція безпеки в життєвий цикл розробки та експлуатації (DevSecOps). Хоча концепція “security as code” активно розвивається, у більшості компаній перевірки безпеки здійснюються вже після розгортання інфраструктури. Це призводить до ситуацій, коли потенційно небезпечні конфігурації потрапляють у продакшн, і лише потім виявляються системами моніторингу. Таким чином, DevSecOps ще не став повністю інтегрованим процесом, а радше існує як додатковий шар контролю, що не завжди впливає на поведінку розробників чи адміністраторів.

Третім слабким місцем є залежність від ручних процедур реагування та інтерпретації результатів перевірок. Навіть сучасні системи на кшталт AWS Foundational Security Best Practices або CIS Benchmark Scanners потребують участі фахівця для аналізу “findings” і визначення їхньої критичності. У великих інфраструктурах це призводить до накопичення сотень нерозглянутих сповіщень

і зниження ефективності команд безпеки. У науковій літературі (Luo et al., 2023; Javed et al., 2022) зазначається, що у понад 40 % організацій понад половина автоматичних попереджень залишається без реагування через відсутність контексту чи автоматизованої пріоритизації. Таким чином, безперервний моніторинг без механізму оцінювання ризиків не забезпечує реального підвищення безпеки.

Ще однією проблемою є недостатня узгодженість між стандартами та реальними технічними перевірками. Стандарти на кшталт NIST SP 800-53 або ISO/IEC 27017 описують загальні вимоги, але не надають конкретних алгоритмів перевірки, тоді як технічні фреймворки (CIS, FSBP) часто не враховують контекст ризиків і взаємозв'язки між ними. У результаті організації змушені самостійно будувати "міст" між політиками та автоматизованими засобами аудиту, що потребує додаткових ресурсів і спеціалізованих знань. Така ситуація створює розрив між теоретичними моделями безпеки й практичними інструментами, які використовуються у щоденній експлуатації.

Проблемою, що залишається поза увагою багатьох досліджень, є відсутність уніфікованої системи метрик для оцінювання ефективності безпеки. У більшості випадків організації покладаються на якісні оцінки ("compliant" / "non-compliant") або на кількість усунутих вразливостей, не враховуючи вагу кожного ризику, середній час виявлення (MTTD) чи відновлення (MTTR). Це унеможливлює кількісне порівняння стану безпеки між середовищами та не дозволяє об'єктивно оцінювати вплив впроваджених заходів. Питання формалізації метрик безпеки у хмарі наразі залишається відкритим у наукових публікаціях і є перспективним напрямом подальших досліджень.

Нарешті, варто відзначити обмежену доступність відкритих репрезентативних даних для тестування й верифікації методів виявлення загроз. Через політику конфіденційності провайдерів більшість досліджень спирається або на симульовані середовища, або на неповні логи, що не дозволяє об'єктивно порівняти точність і ефективність різних підходів. Це обмежує можливості академічної спільноти у створенні універсальних моделей виявлення вразливостей.

Таким чином, основними недоліками сучасних підходів є:

- відсутність єдиної системи контролю для мультихмарних середовищ;
- слабка інтеграція безпеки у DevSecOps-цикли;
- перевантаження команд великою кількістю "сирих" сповіщень без автоматизованої оцінки ризиків;
- розрив між теоретичними стандартами й практичними перевірками;
- відсутність кількісних показників ефективності;
- нестача достовірних даних для навчання та тестування аналітичних моделей.

Усі перелічені проблеми свідчать про потребу у створенні уніфікованої, автоматизованої методики, здатної об'єднати принципи міжнародних стандартів із технічними засобами перевірки, а також надати зрозумілі, вимірювані показники безпеки. Саме ці завдання будуть розглянуті у другому розділі, де

буде обгрунтовано критерії оцінювання ефективності засобів захисту та запропоновано методологію формалізації показників безпеки для хмарних середовищ.

Висновки до розділу 1

У першому розділі проведено комплексний аналіз сучасного стану хмарної безпеки, який охоплює як теоретичні аспекти, так і практичні тенденції у сфері захисту інформації. Узагальнення наукових джерел, стандартів і звітів провідних аналітичних організацій дозволило сформувати цілісне уявлення про ключові виклики та обмеження, що визначають рівень безпеки хмарних середовищ.

Насамперед було з’ясовано, що основою побудови захищених хмарних систем залишаються класичні принципи конфіденційності, цілісності та доступності, доповнені сучасними концепціями підзвітності, керування ідентичностями та принципом найменших привілеїв. Саме ці принципи визначають архітектурні та організаційні рішення, що забезпечують надійність хмарних сервісів.

Аналіз статистичних звітів Cloud Security Alliance, IBM X-Force та ENISA показав, що понад половина інцидентів у хмарних середовищах пов’язана не з вразливостями самих провайдерів, а з помилками конфігурацій (misconfigurations), неналежним управлінням доступом (IAM) та відсутністю повноцінного моніторингу. Приклади інцидентів у компаніях Capital One, Marriott, UniSuper, American Airlines та SingHealth підтвердили ці тенденції й продемонстрували, що навіть організації з розвиненими IT-службами не застраховані від людських помилок та недоліків у політиках доступу.

Детальний огляд міжнародних стандартів (CIS Benchmarks, NIST SP 800-53, ISO/IEC 27017/27018, CSA CCM, AWS Foundational Security Best Practices) показав, що кожен із них вирішує лише частину завдань безпеки: одні акцентують на управлінських процесах, інші - на технічних перевірках. Їхня інтеграція дає змогу створити більш повну модель контролю ризиків, однак на практиці така узгодженість досягається не завжди. Відсутність єдиного підходу до мультихмарних середовищ залишається однією з ключових проблем.

Огляд академічних праць 2019–2025 рр. засвідчив перехід досліджень від описових моделей до практичних, автоматизованих підходів - таких як Cloud Security Posture Management, DevSecOps та policy-as-code. Науковці пропонують методи формальної перевірки конфігурацій, мінімізації прав доступу, динамічного вимірювання рівня безпеки й раннього виявлення аномалій. Проте водночас виявлено низку прогалин: відсутність єдиних метрик ефективності, обмежена доступність реальних даних для тестування, розрив між високорівневими стандартами та технічними інструментами.

Таким чином, перший розділ дозволив:

- визначити основні загрози та уразливості хмарних середовищ;
- окреслити модель сучасних засобів захисту і проаналізувати їх ефективність;
- виявити системні недоліки існуючих підходів - фрагментарність, недостатню автоматизацію, відсутність кількісної оцінки ризиків;
- сформувати наукове та практичне підгрунтя для розроблення власних рекомендацій і методики оцінювання.

Отже, перший розділ підготував основу для подальшого дослідження. У другому розділі буде обгрунтовано систему показників і критеріїв, за якими можна кількісно оцінювати ефективність засобів захисту в хмарному середовищі, що стане фундаментом для подальшого проєктування та експериментальної перевірки розробленої утиліти.

Розділ 2. Обгрунтування показників і критеріїв оцінювання ефективності засобів захисту в хмарному середовищі

Забезпечення інформаційної безпеки у хмарному середовищі потребує не лише впровадження технічних і організаційних заходів, а й постійного вимірювання їхньої результативності. В умовах динамічності хмарних інфраструктур, автоматизації процесів і великої кількості змін у конфігураціях саме система показників і критеріїв дозволяє об’єктивно оцінювати рівень захисту та ефективність вжитих заходів.

Після аналізу сучасних підходів (розділ 1) було встановлено, що більшість існуючих методик концентрується на перевірці відповідності політикам, але рідко надає кількісну оцінку впливу тих чи інших рішень. Тому виникає потреба у формалізації метрик безпеки, які можна використовувати для порівняння різних систем захисту, моніторингу змін стану безпеки та вимірювання прогресу впроваджених рекомендацій.

Метою цього розділу є обгрунтування системи критеріїв та показників, за допомогою яких можна оцінити ефективність засобів захисту інформації в хмарному середовищі. Для цього розглядаються теоретичні основи поняття ефективності, аналізуються існуючі стандартизовані підходи до оцінювання (зокрема - NIST SP 800-55, ISO/IEC 27004), а також пропонуються групи показників - технічні, організаційні, часові та ризик-орієнтовані. У підсумку формується узгоджена методика, яка стане основою для подальшого аналізу та експериментальної перевірки в наступних розділах.

2.1. Теоретичні засади оцінювання ефективності систем захисту

2.1.1. Поняття “ефективності” в контексті інформаційної безпеки

У системах захисту інформації термін ефективність означає ступінь досягнення запланованого рівня безпеки при раціональному використанні ресурсів. Інакше кажучи, ефективність відображає, наскільки впроваджені заходи реально знижують рівень ризику до прийнятного значення з точки зору організації. Це поняття має як кількісний, так і якісний вимір, оскільки охоплює технічні, організаційні та економічні аспекти безпеки.

У нормативних джерелах ефективність системи інформаційної безпеки трактується як «здатність забезпечувати виконання вимог конфіденційності, цілісності, доступності та підзвітності інформації при мінімальних витратах ресурсів» (ISO/IEC 27004:2016). У цьому стандарті ефективність безпеки розглядається не лише як факт наявності певних контролів, а як ступінь досягнення їхніх цілей, що вимірюється через відповідні метрики та показники результативності (Performance Indicators).

У публікації NIST SP 800-55 Rev. 1 «Performance Measurement Guide for Information Security» підкреслюється, що ефективність - це співвідношення між очікуваними результатами безпекових заходів і фактичними показниками ризику або інцидентності. Тобто, система вважається ефективною, якщо після впровадження контролів рівень ризику або кількість порушень зменшується, а процеси виявлення та реагування стають швидшими. Цей підхід передбачає безперервне вимірювання, що є особливо важливим для динамічних хмарних середовищ.

У контексті хмарних технологій ефективність засобів захисту слід розглядати багатовимірно. З одного боку, вона відображає ступінь зниження технічних вразливостей - наприклад, кількість помилкових конфігурацій, відсутність надлишкових прав, своєчасність оновлень. З іншого боку, ефективність включає організаційний вимір - наскільки чітко визначені політики доступу, автоматизовані процеси перевірки та інтегрована безпека в DevOps-цикл. Третім аспектом є економічна ефективність, яка оцінює співвідношення між витратами на впровадження контролів і скороченням потенційних втрат від інцидентів.

Таким чином, у хмарному середовищі ефективність засобів захисту можна визначити як сукупну здатність системи безпеки підтримувати прийнятний рівень ризику при оптимальному використанні ресурсів, враховуючи швидкість реагування на загрози, точність виявлення порушень і ступінь автоматизації процесів контролю. Вимірювання ефективності стає не лише інструментом оцінки поточного стану, а й ключовим елементом управління безпекою, що дозволяє приймати обґрунтовані рішення щодо вдосконалення політик, засобів і процедур.

2.1.2. Відмінність між якісними та кількісними показниками безпеки

Оцінювання ефективності системи захисту інформації неможливе без визначення показників, які відображають її стан. У міжнародній практиці такі показники поділяються на якісні (qualitative) та кількісні (quantitative) - кожен тип має своє призначення, переваги та обмеження.

Якісні показники використовуються тоді, коли оцінку неможливо або недоцільно подати у числовій формі. Вони базуються на експертних судженнях, порівнянні із встановленими нормами чи стандартами, а також на результатах аудитів. Прикладами якісних метрик є:

- наявність або відсутність затвердженої політики безпеки;
- ступінь зрілості процесу управління ризиками;
- виконання вимог стандартів ISO/IEC 27001 чи NIST SP 800-53;
- рівень обізнаності персоналу щодо політик безпеки;
- рівень інтеграції принципів DevSecOps у процес розробки.

Такі показники добре відображають загальний стан управління безпекою, проте не дозволяють виміряти фактичний ефект від конкретних заходів. Вони важливі для стратегічного рівня управління (CISO, аудитори, комплаєнс-офіцери), але мають обмежену цінність при технічному аналізі.

Кількісні показники, навпаки, відображають безпеку через вимірювані, числові величини. Вони дозволяють проводити об’єктивні порівняння у часі, між середовищами або різними наборами контролів.

Основними прикладами таких метрик є:

- MTTD (Mean Time to Detect) - середній час виявлення інциденту;
- MTTR (Mean Time to Respond / Recover) - середній час реагування або відновлення;
- Compliance Rate - частка ресурсів, що відповідають політикам безпеки (%);
- Remediation Rate - частка усунутих вразливостей за певний період;
- Coverage Ratio - частка охоплених перевіркою сервісів чи конфігурацій;
- False Positive Rate - частота хибних спрацювань засобів моніторингу;
- Security Posture Score - інтегральний показник стану безпеки за зваженими критеріями.

У документі NIST SP 800-55 Rev.1 “Performance Measurement Guide for Information Security” кількісні метрики класифікуються за трьома рівнями:

1. Implementation metrics - відображають ступінь реалізації контролів (наприклад, кількість користувачів із ввімкненим MFA);
2. Effectiveness metrics - показують фактичний результат (зменшення кількості інцидентів, швидше виявлення тощо);
3. Efficiency metrics - оцінюють співвідношення результату до витрат (наприклад, вартість усунення інциденту проти вартості впровадження контролю).

У стандарті ISO/IEC 27004:2016 “Information Security Management - Measurement” підкреслюється, що якісні та кількісні показники мають застосовуватись у поєднанні: якісні метрики формують контекст і дозволяють оцінити зрілість процесів, а кількісні - забезпечують вимірюваність та

можливість прийняття управлінських рішень на основі даних (evidence-based security management).

Для хмарних середовищ особливо важливо досягнути балансу між цими двома типами показників. З одного боку, кількісні метрики дозволяють автоматизувати аналіз (через інструменти CSPM, SIEM, CloudTrail, Security Hub), з іншого - якісні показники відображають організаційну готовність, культуру безпеки та ступінь зрілості процесів, які не можна виміряти автоматично.

Таким чином, ефективна система оцінювання має поєднувати:

- кількісні метрики для моніторингу технічного стану безпеки (час, відсотки, інтенсивність порушень);
- якісні метрики для оцінки управлінських і процедурних аспектів (зрілість процесів, політики, відповідальність).

Це поєднання дозволяє створити повну картину ефективності - від технічного рівня конфігурацій і подій до стратегічного рівня управління ризиками та безпековими інвестиціями.

2.1.3. Підходи до вимірювання ефективності: ризик-орієнтований, процесний та технічний

Вимірювання ефективності засобів захисту інформації може здійснюватися різними методами залежно від цілей аналізу, типу організації та зрілості її системи безпеки. У науковій та практичній літературі виділяють три основні підходи до оцінювання ефективності: ризик-орієнтований, процесний та технічний. Кожен із них розглядає безпеку під власним кутом - від управління ризиками до аналізу конкретних технічних показників.

Ризик-орієнтований підхід

Ризик-орієнтований підхід (risk-based approach) є основоположним у стандартах NIST SP 800-30, ISO/IEC 27005:2022 та ENISA Risk Management Framework. Його сутність полягає у тому, що ефективність засобів безпеки оцінюється через ступінь зниження ризику до прийнятного рівня.

Інакше кажучи, не просто перевіряється наявність контролю, а вимірюється, наскільки він зменшує імовірність або наслідки інциденту.

У хмарному середовищі ризик-орієнтований підхід передбачає такі кроки:

1. Визначення критичних активів (наприклад, S3-сховищ, баз даних, IAM-ролей).
2. Оцінка ризиків для кожного активу за формулою Risk = Likelihood × Impact.
3. Вибір контрольних заходів та оцінка їх впливу на ризик.
4. Вимірювання залишкового ризику після впровадження засобів захисту.

Наприклад, якщо конфігураційна перевірка знизила кількість публічних S3-бакетів на 80 %, можна вважати, що ризик несанкціонованого доступу до даних зменшено пропорційно. Такий підхід надає змогу приймати управлінські

рішення, грунтуючись на реальному ефекті контролів, а не лише на формальному дотриманні політик.

Процесний підхід

Процесний або management-based approach грунтується на оцінюванні зрілості та стабільності процесів управління безпекою. Цей підхід характерний для стандартів ISO/IEC 27004:2016, COBIT 5, CMMI-SVC та моделей зрілості (Security Maturity Models).

Ефективність у цьому контексті визначається не лише результатом, а й здатністю організації системно управляти безпекою, тобто мати політики, процедури, відповідальність і контроль циклу PDCA (Plan – Do – Check – Act).

Прикладом застосування процесного підходу в хмарі є оцінка рівня інтеграції DevSecOps. Якщо безпека перевіряється лише після деплою, процес вважається низькозрілим; якщо політики перевіряються автоматично на етапі CI/CD - середньозрілим; якщо результати перевірок використовуються для автоматичного виправлення конфігурацій (auto-remediation) - високозрілим.

Таким чином, основним показником виступає не кількість знайдених проблем, а здатність організації їх системно виявляти, виправляти та запобігати повторенню.

Процесний підхід особливо важливий у великих організаціях, де технічна ефективність може бути високою локально, але відсутність стандартизованих процедур і єдиних ролей призводить до хаотичного управління безпекою. У таких випадках саме оцінка зрілості процесів дозволяє визначити слабкі місця системи.

Технічний підхід

Технічний або metric-based approach зосереджується на кількісних показниках, які можна безпосередньо виміряти в хмарному середовищі. Це найбільш конкретний і формалізований метод оцінювання ефективності, що широко застосовується у сучасних системах Cloud Security Posture Management (CSPM), Security Information and Event Management (SIEM) та Vulnerability Management (VM).

Основними технічними метриками є:

- кількість активних вразливостей;
- відсоток ресурсів, що відповідають вимогам політик (compliance rate);
- середній час виправлення вразливості (MTTR);
- середній час між інцидентами (MTBF);
- частка автоматизованих перевірок у загальному обсязі контролів;
- показники навантаження системи моніторингу (event/sec, log volume).

У контексті хмарних платформ технічний підхід має суттєву перевагу - можливість автоматизації. Наприклад, інструменти типу AWS Security Hub або Prisma Cloud можуть автоматично обчислювати compliance-score для кожного акаунта й надавати зведену оцінку рівня безпеки у відсотках. Однак технічний підхід не враховує контекст бізнесу: система може показувати високу кількість сповіщень, але їхній фактичний ризик буде низьким.

Таблиця 2. Порівняльна характеристика підходів

Підхід	Основна мета	Приклади показників	Переваги	Обмеження
Ризик-орієнтований	Зниження ризиків до прийнятного рівня	Імовірність інциденту, вплив, залишковий ризик	Орієнтований на цінність активів, пріоритизація	Складність кількісної оцінки
Процесний	Оцінка зрілості процесів і політик	Рівень інтеграції DevSecOps, наявність процедур, ролей	Дає системне бачення, підходить для аудитів	Менш придатний для автоматизації
Технічний	Вимірюваність контролів і подій	MTTD, MTTR, compliance rate, false positive rate	Об’єктивність, автоматизація	Відірваність від бізнес-контексту

У більшості сучасних моделей безпеки застосовується комбінований підхід, який поєднує ризикову пріоритизацію, процесну зрілість та технічну вимірюваність. Саме така інтеграція є особливо актуальною для хмарних середовищ, де безпека має динамічний характер і вимагає постійного перерахунку показників у реальному часі.

У наступних підрозділах буде запропоновано систему критеріїв, яка синтезує ці три підходи та забезпечує кількісну основу для подальшої оцінки ефективності засобів захисту.

2.1.4. Зв’язок між рівнем безпеки, ризиком та вартістю реалізації контролів

Оцінювання ефективності системи інформаційної безпеки неможливе без урахування взаємозв’язку між трьома ключовими елементами - рівнем безпеки, ризиком та вартістю впровадження заходів захисту. У сучасних концепціях управління безпекою (зокрема, NIST Risk Management Framework, ISO/IEC 27005, ENISA Guidelines) ці фактори розглядаються як взаємозалежні складові єдиної системи прийняття рішень.

Рівень безпеки (Security Level) визначається сукупністю впроваджених заходів, які зменшують імовірність реалізації загроз і пом’якшують наслідки можливих інцидентів. Він відображає стан системи у конкретний момент часу та може бути представлений у вигляді інтегрального показника - наприклад, Security Posture Score або Compliance Index. Високий рівень безпеки, однак, не

завжди означає оптимальність: досягти максимальної захищеності можливо лише за умови надмірних витрат або обмеження функціональності системи.

Ризик (Risk) у цьому контексті є мірою невизначеності, що поєднує ймовірність інциденту та ступінь його впливу. Як правило, він визначається за формулою:

$$\boldsymbol{R = P * I}$$

де P - імовірність настання події, а I - потенційні збитки (фінансові, репутаційні, операційні).

Ефективність заходів безпеки можна інтерпретувати як ступінь зниження ризику після впровадження контролів, тобто різницю між первинним ризиком (R_0) і залишковим ризиком (R_r):

$$\boldsymbol{E = \frac{R_0 - R_r}{R_0}}$$

де E - коефіцієнт ефективності безпеки.

Вартість реалізації заходів (Cost of Control Implementation) включає не лише прямі фінансові витрати, але й непрямі - зниження продуктивності, час, людські ресурси та вплив на зручність користувачів.

Згідно з моделлю Cost–Benefit Analysis (CBA), описаною у NIST SP 800-30 та ISO/IEC 27005, захід вважається доцільним, якщо очікуване зменшення ризику перевищує витрати на його реалізацію:

$$\boldsymbol{C_{control} \leq R_0 - R_r}$$

Це означає, що система безпеки має бути економічно обгрунтованою: надмірна кількість контролів може не лише не підвищити загальну безпеку, а й ускладнити адміністрування, створити нові точки відмови та знизити ефективність роботи персоналу.

У хмарному середовищі цей баланс є особливо важливим, оскільки витрати на безпеку прямо залежать від обсягу використаних ресурсів (логування, сканування, моніторинг, шифрування). Наприклад, постійне логування всіх API-викликів у AWS CloudTrail суттєво підвищує видимість, але одночасно збільшує витрати на зберігання логів і може створювати надлишкове навантаження на обробку даних.

Тому доцільність заходів повинна оцінюватися з урахуванням співвідношення між додатковими витратами та реальним зниженням ризику.

У практиці управління хмарною безпекою застосовується концепція оптимального рівня безпеки (Optimal Security Level) - стану, коли сукупна вартість заходів і очікуваних збитків мінімальна. Цей рівень досягається в точці рівноваги між інвестиціями в безпеку та залишковим ризиком. Графічно залежність можна подати як U-подібну криву: з ростом витрат ризик спочатку

різко зменшується, потім стабілізується, а надалі зниження ризику стає незначним порівняно з витратами.

Таким чином, ефективність системи безпеки визначається не лише кількістю чи якістю впроваджених контролів, а й оптимальністю їхнього поєднання, що забезпечує прийнятний рівень ризику при мінімально необхідних витратах.

У хмарному середовищі це означає необхідність використання метрик, які враховують взаємозв'язок між ризиком, витратами та технічними показниками, наприклад:

- Cost per Mitigated Risk - вартість зниження одиниці ризику;
- Return on Security Investment (ROSI) - співвідношення між економією від запобігання інцидентам і витратами на заходи безпеки;
- Residual Risk Index - рівень залишкового ризику після впровадження контролів.

Застосування цих підходів дозволяє не лише оцінити поточну ефективність системи, але й обгрунтувати вибір пріоритетних заходів у майбутньому. Саме тому в наступному розділі 2.2 буде сформовано конкретні критерії оцінювання ефективності засобів захисту інформації, що враховують як ризикову, так і економічну складову.

2.2. Критерії оцінювання ефективності засобів захисту інформації

2.2.1. Загальні критерії оцінювання ефективності

Для того щоб об'єктивно оцінити ефективність системи захисту інформації, необхідно визначити набір універсальних критеріїв, які можна застосувати незалежно від конкретної технологічної реалізації. Такі загальні критерії формують основу будь-якої методики оцінювання - від державних стандартів до корпоративних політик безпеки.

У міжнародній практиці найчастіше використовуються чотири ключові ознаки: відповідність (compliance), повнота (completeness), своєчасність (timeliness) та узгодженість (consistency).

Відповідність (Compliance)

Критерій відповідності відображає ступінь узгодження системи безпеки з вимогами нормативних документів, політик, галузевих стандартів або внутрішніх регламентів організації.

У практиці інформаційної безпеки відповідність розглядається як первинна умова ефективності - якщо система не відповідає базовим вимогам (наприклад, NIST, ISO/IEC 27001, CIS Benchmarks, GDPR), вона не може вважатися захищеною незалежно від кількості встановлених контролів.

У хмарному середовищі цей критерій набуває особливої ваги через динамічність інфраструктури: ресурси створюються, змінюються та

видаляються автоматично. Тому перевірка відповідності має бути безперервною й автоматизованою. Наприклад, інструменти AWS Config або Azure Policy можуть постійно контролювати відповідність налаштувань вимогам CIS або ISO 27017.
Метрика для цього критерію зазвичай виражається у відсотках:

$$\boldsymbol{Compliance\ Rate} = \frac{\boldsymbol{N_{compliant}}}{\boldsymbol{N_{total}}} * \mathbf{100}\%$$

Де $N_{compliant}$ - кількість ресурсів, що відповідають політикам, а N_{total}- загальна кількість перевірених ресурсів.

Повнота (Completeness)

Повнота визначає ступінь охоплення системою контролю всіх релевантних об’єктів і процесів, які можуть впливати на рівень безпеки. Іншими словами, навіть якщо окремі елементи добре захищені, система не може вважатися ефективною, якщо частина ресурсів не охоплена перевірками або політиками.

У хмарному середовищі критично важливо враховувати, що дані можуть розміщуватись у різних регіонах і акаунтах, а політики часто застосовуються вибірково. Наприклад, аудит лише S3-бакетів без перевірки IAM-ролей або VPC-конфігурацій не дає повного уявлення про стан безпеки.
Показник повноти може бути представлений формулою:

$$\boldsymbol{Coverage\ rate} = \frac{\boldsymbol{N_{controlled}}}{\boldsymbol{N_{existing}}} * \mathbf{100}\%$$

де $N_{controlled}$- кількість ресурсів, що перебувають під контролем або моніторингом, а $N_{existing}$ - фактична кількість об’єктів у системі.

Таким чином, повнота демонструє не стільки точність окремих перевірок, скільки рівень “прозорості” середовища, тобто наскільки повно охоплено весь обсяг потенційних ризиків.

Своєчасність (Timeliness)

Своєчасність характеризує здатність системи безпеки виявляти, повідомляти та усувати порушення у прийнятні часові межі. Цей критерій напряму пов’язаний із показниками MTTD (Mean Time to Detect) та MTTR (Mean Time to Respond), що використовуються у багатьох міжнародних стандартах, зокрема NIST SP 800-61 (Computer Security Incident Handling Guide).

У хмарних середовищах своєчасність має подвійне значення:

- по-перше, через динамічність ресурсів важливо, щоб засоби моніторингу оперативно реагували на будь-які зміни конфігурацій;

- по-друге, час реагування напряму впливає на наслідки інциденту - затримка навіть у кілька годин може призвести до компрометації великого обсягу даних.

Для вимірювання цього критерію можуть застосовуватись такі метрики:

$$\boldsymbol{MTTD} = \frac{\sum \boldsymbol{t_{detect}}}{\boldsymbol{n}}; \quad \boldsymbol{MTTR} = \frac{\sum \boldsymbol{t_{response}}}{\boldsymbol{n}}$$

де t_{detect} - час від моменту виникнення інциденту до його виявлення, $t_{response}$ - час до усунення, а n - кількість інцидентів за певний період.

Чим менші значення цих показників, тим вищою є своєчасність системи реагування. На практиці досягнення прийнятного рівня своєчасності передбачає автоматизацію сповіщень, інтеграцію SIEM/SOAR-платформ і використання безперервного моніторингу у реальному часі.

Узгодженість (Consistency)

Критерій узгодженості відображає ступінь єдності принципів, політик та технічних контролів у межах усієї системи безпеки. Вона означає, що однакові вимоги застосовуються до всіх середовищ, а зміни впроваджуються системно, без суперечностей між різними рівнями управління.

Висока узгодженість забезпечує прогнозовану поведінку системи: якщо правила шифрування або керування ключами діють у всіх регіонах і акаунтах однаково, ризик людської помилки суттєво зменшується.

У багатохмарних середовищах цей критерій є особливо складним - різні провайдери (AWS, Azure, GCP) мають відмінні політики й механізми безпеки, тому досягнення узгодженості потребує уніфікації політик та централізованого управління.

Показник узгодженості може бути оцінений як частка конфігурацій, що відповідають корпоративним шаблонам (наприклад, через Policy-as-Code або Infrastructure-as-Code):

$$\boldsymbol{Consistency\ Rate} = \frac{\boldsymbol{N_{aligned}}}{\boldsymbol{N_{checked}}} * \mathbf{100}\%$$

Загальні критерії - відповідність, повнота, своєчасність і узгодженість - утворюють базову основу для оцінювання ефективності будь-якої системи безпеки. Вони дозволяють оцінити не лише технічний стан хмарної інфраструктури, але й організаційну дисципліну та якість управління.

Якщо хоча б один із цих критеріїв має низьке значення, загальна ефективність системи суттєво знижується, навіть за наявності передових технологічних засобів. У подальших підрозділах ці критерії будуть деталізовані через технічні, організаційні, часові та економічні показники, що формують кількісну основу для інтегрального оцінювання рівня безпеки.

2.2.2. Технічні критерії оцінювання ефективності засобів захисту

Технічні критерії відображають кількісну сторону ефективності системи безпеки, тобто наскільки технічні засоби здатні забезпечити виявлення, попередження та усунення порушень безпеки у хмарному середовищі. Вони дозволяють не лише оцінити наявність контролів, а й виміряти їх фактичну результативність у цифрових показниках.

Такі критерії широко використовуються у стандартах NIST SP 800-55, CIS Benchmarks, ISO/IEC 27004, а також у практиці Cloud Security Posture Management (CSPM), Security Information and Event Management (SIEM) та Vulnerability Management (VM).

Рівень автоматизації контролів (Automation Level)

Автоматизація є ключовим чинником технічної ефективності, адже саме вона дозволяє реагувати на загрози у режимі реального часу та мінімізувати людський фактор.

У хмарному середовищі ступінь автоматизації визначається часткою перевірок, які виконуються без участі оператора - наприклад, сканування конфігурацій за допомогою AWS Config, автоматичні тригери безпеки в Lambda, або механізми auto-remediation у Prisma Cloud.

Показник автоматизації можна визначити за формулою:

$$\boldsymbol{Automation\ rate} = \frac{\boldsymbol{N_{automated}}}{\boldsymbol{N_{total}}} * \mathbf{100\%}$$

Де $N_{automated}$ – кількість автоматизованих перевірок, N_{total} – загальна кількість перевірок.

Високе значення цього показника свідчить про зрілість системи та її здатність підтримувати безпеку у масштабі без додаткового навантаження на персонал.

Покриття сервісів та компонентів (Coverage of Services)

Цей критерій показує, наскільки повно засоби захисту охоплюють різні елементи хмарної інфраструктури - обчислювальні ресурси, мережеві конфігурації, сховища, IAM, бази даних, журнали подій тощо.

Для великих організацій критично важливо забезпечити перевірку всіх типів сервісів, незалежно від того, чи це IaaS, PaaS чи SaaS.

Метрика визначається як:

$$\boldsymbol{Services\ coverage\ rate} = \frac{\boldsymbol{N_{covered}}}{\boldsymbol{N_{existing}}} * \mathbf{100\%}$$

де $N_{covered}$ – кількість покритих сервісів, $N_{existing}$ – загальна кількість сервісів.

Недостатнє покриття часто стає причиною “сліпих зон”, де ризики залишаються невиявленими. Наприклад, якщо система моніторить S3 та EC2, але не перевіряє IAM чи CloudTrail, реальна картина безпеки буде неповною.

Точність виявлення порушень (Detection Accuracy)

Точність характеризує здатність системи правильно ідентифікувати реальні інциденти без хибних спрацювань. У науковій літературі вона визначається через показники True Positive Rate (TPR) і False Positive Rate (FPR), які разом формують Precision та Recall системи моніторингу.

$$\boldsymbol{Precision} = \frac{\boldsymbol{TP}}{\boldsymbol{TP + FP}};\ \boldsymbol{Recall} = \frac{\boldsymbol{TP}}{\boldsymbol{TP + FN}}$$

де TP - кількість правильних виявлень, FP - хибні спрацювання, FN - пропущені інциденти.

У хмарних системах висока точність означає, що засіб безпеки здатен адекватно розрізняти нормальну активність від шкідливої, не створюючи перевантаження операційного центру SOC.

Ефективність усунення вразливостей (Remediation Efficiency)

Оцінює, наскільки швидко та якісно усуваються знайдені проблеми. Вона визначається як відношення кількості усунутих порушень до загальної кількості виявлених протягом певного періоду:

$$\boldsymbol{Remediation\ rate} = \frac{\boldsymbol{N_{fixed}}}{\boldsymbol{N_{found}}} * \mathbf{100\%}$$

де N_{fixed} – кількість вирішених вразливостей, N_{found} – загальна кількість знайдених вразливостей

Висока швидкість усунення свідчить про налагоджений процес взаємодії між командами DevOps і безпеки, а низька - про накопичення технічного боргу та ризик повторних інцидентів.

Цей показник часто корелює з рівнем автоматизації: організації, що використовують auto-remediation або policy-as-code, зазвичай мають на 30–50 % вищу ефективність виправлення конфігурацій.

Інтегральний показник технічної ефективності (Security Posture Score)

Для комплексної оцінки технічного стану можна використовувати інтегральний показник - Security Posture Score, який об’єднує попередні критерії у зважену формулу:

$$\boldsymbol{SPS} = \boldsymbol{\omega_1 A + \omega_2 C + \omega_3 P + \omega_4 R}$$

де A - рівень автоматизації, C - покриття, P - точність виявлення, R - швидкість усунення, w_1…w_4 - вагові коефіцієнти, визначені відповідно до пріоритетів організації.

Такий підхід застосовується в системах типу AWS Security Hub, Azure Defender чи Google Security Command Center, які розраховують інтегральний бал безпеки на основі набору контрольних перевірок.

Узагальнюючи, технічні критерії дають змогу кількісно оцінити стан безпеки хмарної інфраструктури та відстежувати зміни у динаміці. Вони формують основу для автоматизованого аудиту і дозволяють інтегрувати результати оцінювання у звіти ризик-менеджменту.

У наступному підпункті буде розглянуто організаційні критерії, які доповнюють технічні показники, охоплюючи процеси управління, політики, розподіл ролей і рівень зрілості системи безпеки.

2.2.3. Організаційні критерії оцінювання ефективності засобів захисту

Організаційні критерії відображають зрілість управління інформаційною безпекою та ступінь інтеграції безпеки у внутрішні процеси організації. Якщо технічні критерії показують, як працюють засоби захисту, то організаційні визначають, наскільки системно організація підходить до їх застосування, підтримки та вдосконалення.

У міжнародних стандартах ISO/IEC 27001, ISO/IEC 27004, NIST SP 800-53, а також у рекомендаціях ENISA підкреслюється, що ефективність системи безпеки неможливо забезпечити виключно технічними засобами - вона залежить від організаційної культури, чіткості ролей і наявності формалізованих процесів.

Зрілість процесів управління безпекою (Security Maturity)

Критерій зрілості відображає наскільки послідовно та системно організація впроваджує заходи безпеки. Для цього часто застосовують моделі на кшталт CMMI (Capability Maturity Model Integration), COBIT 5 або Security Maturity Model (SMM).

Залежно від рівня зрілості, організація може перебувати на різних стадіях - від реактивного до проактивного управління:

Таблиця 3. Рівні зрілості організацій

Рівень	Характеристика
1. Початковий	Реакція на інциденти без формальних процедур
2. Повторюваний	Є базові політики, але контроль несистемний
3. Визначений	Процеси документовані, контроль відбувається регулярно
4. Керований	Метрики відстежуються, проводиться аналіз ефективності

Рівень	Характеристика
5. Оптимізований	Постійне вдосконалення, автоматизація та інтеграція DevSecOps

У хмарному контексті високий рівень зрілості означає, що організація має централізоване управління акаунтами, стандартизовані політики для всіх середовищ, автоматизовані перевірки конфігурацій та узгоджений цикл управління ризиками.

Розподіл ролей та відповідальності (Governance and Accountability)

Ефективність системи безпеки значною мірою залежить від чіткого визначення ролей і зон відповідальності. У хмарних середовищах це особливо важливо через модель “спільної відповідальності” (Shared Responsibility Model), коли частина контролів належить провайдеру, а частина - користувачу.

Організаційний критерій у цьому випадку оцінює:

- наявність чітко визначених посадових інструкцій у сфері кібербезпеки;
- розподіл прав доступу до хмарних ресурсів (адміністратори, розробники, аудитори);
- наявність процесів контролю змін і затвердження політик;
- ефективність внутрішніх комунікацій між командами DevOps, SecOps і Compliance.

Наявність формалізованих ролей (наприклад, Cloud Security Officer, IAM Administrator, Incident Manager) забезпечує підзвітність і дає змогу швидко реагувати на відхилення або інциденти.

Інтеграція безпеки в життєвий цикл розробки (DevSecOps Integration)

Сучасні підходи до оцінювання ефективності враховують ступінь інтеграції безпеки у процеси розробки, тестування та експлуатації. У традиційних середовищах перевірка безпеки відбувається після релізу, що призводить до затримок і збільшення ризику. Натомість DevSecOps-практики передбачають перевірку політик безпеки вже на етапі CI/CD (shift-left security).

Показниками ефективної інтеграції є:

- наявність автоматичних перевірок політик у пайплайнах CI/CD (наприклад, Checkov, OPA, tfsec);
- кількість уразливостей, виявлених до етапу продакшену;
- відсоток релізів, що пройшли без порушення вимог безпеки;
- використання “policy-as-code” для уніфікації контролів.

Високий рівень інтеграції DevSecOps зменшує середній час виправлення конфігурацій (MTTR) і забезпечує стабільність безпеки в умовах частих змін хмарної інфраструктури.

Навчання та обізнаність персоналу (Awareness and Training)

Жодна технічна система не може бути ефективною без належної підготовки користувачів. Тому важливим організаційним критерієм є рівень обізнаності персоналу у сфері безпеки.

Цей показник оцінюється за такими параметрами:

- регулярність навчань і тестування знань (наприклад, simulated phishing або cloud security workshops);
- відсоток працівників, які пройшли сертифікацію або тренінги (AWS Certified Security, CompTIA Security+, CISSP);
- кількість інцидентів, спричинених людським фактором (наприклад, помилки конфігурації або неправильне керування доступом).

Згідно з дослідженнями IBM X-Force (2024), до 80 % інцидентів у хмарі мають у своїй основі людський фактор, тому навчання та культура безпеки прямо впливають на загальний рівень захисту.

Управління інцидентами та безперервність бізнесу

Цей критерій оцінює готовність організації до реагування на інциденти та відновлення після них. Згідно зі стандартами ISO/IEC 22301 та NIST SP 800-61, ефективна система безпеки має включати документовані процедури реагування, резервні копії, плани аварійного відновлення (Disaster Recovery Plan) та тестування цих процедур.

У хмарних середовищах це означає:

- наявність автоматичних резервних копій у різних регіонах (cross-region replication);
- регулярні тести відновлення (DR tests);
- середній час відновлення після збою (Mean Time to Restore).

Висока організаційна ефективність передбачає, що навіть у випадку серйозного інциденту система може швидко відновити роботу без значних втрат даних чи часу.

Узагальнюючи, організаційні критерії дозволяють оцінити не лише технічну ефективність засобів безпеки, але й здатність організації підтримувати стабільну, контрольовану та керовану систему захисту. Високий рівень організаційної зрілості гарантує, що навіть за зміни технологій або персоналу основні принципи безпеки залишаються сталими.

У наступному підпункті буде розглянуто часові та експлуатаційні критерії, які дозволяють оцінити динамічні характеристики системи - швидкість виявлення, реагування та відновлення після інцидентів.

2.2.4. Часові та експлуатаційні критерії оцінювання ефективності

Часові та експлуатаційні критерії відображають динамічну ефективність системи захисту, тобто її здатність своєчасно виявляти, реагувати та відновлюватись після інцидентів. У хмарних середовищах, де події відбуваються у режимі реального часу, ці показники мають вирішальне значення для збереження безперервності роботи сервісів.

Основними часовими метриками є:
- MTTD (Mean Time to Detect) - середній час виявлення інциденту. Високе значення цього показника свідчить про затримку в моніторингу або відсутність належної автоматизації.
- MTTR (Mean Time to Respond / Recover) - середній час реагування або відновлення. Скорочення MTTR досягається завдяки використанню автоматичних сценаріїв усунення (auto-remediation) та інтеграції SIEM/SOAR.
- MTBF (Mean Time Between Failures) - середній час між відмовами системи, який характеризує стабільність роботи сервісів.

До експлуатаційних критеріїв належать також:
- Uptime / Availability (%) - частка часу, коли система залишається працездатною відповідно до угоди про рівень обслуговування (SLA). Для критичних сервісів хмарні провайдери зазвичай гарантують доступність 99,9–99,99 %.
- Change Latency - час, необхідний для застосування змін без порушення безпеки чи доступності системи.
- Recovery Point Objective (RPO) та Recovery Time Objective (RTO) - показники, що визначають максимальні межі втрати даних і часу відновлення у випадку аварії.

У контексті хмарної інфраструктури оптимальні часові характеристики свідчать про високу оперативність, стійкість та узгодженість дій команди безпеки. Зменшення MTTD і MTTR є одним із найнадійніших індикаторів ефективності, оскільки безпосередньо впливає на мінімізацію наслідків інцидентів.

Таким чином, часові та експлуатаційні критерії формують операційну складову ефективності - вони показують, наскільки швидко і стабільно система реагує на загрози, підтримуючи безперервність бізнес-процесів.

2.2.5. Економічні критерії оцінювання ефективності засобів захисту

Економічні критерії дозволяють оцінити ефективність системи безпеки з точки зору раціонального використання ресурсів та співвідношення між витратами на впровадження заходів і досягнутим рівнем захисту.

У сучасних умовах, коли хмарна інфраструктура передбачає оплату за фактичне споживання ресурсів, фінансовий аспект безпеки стає не менш важливим, ніж технічний чи організаційний.

Метою економічного оцінювання є визначення оптимального балансу між інвестиціями в безпеку та зменшенням ризику інформаційних втрат. Надмірні витрати на засоби захисту можуть призвести до неефективного використання бюджету, тоді як недостатні - до потенційних фінансових збитків унаслідок інцидентів.

Витрати на впровадження та підтримку засобів безпеки

Першим і найочевиднішим економічним критерієм є сукупна вартість власності (Total Cost of Ownership, TCO) системи безпеки. Вона охоплює як прямі, так і непрямі витрати, зокрема:

- вартість ліцензій, підписок або сервісів CSPM / SIEM / SOC;
- витрати на обчислювальні ресурси (логування, моніторинг, резервування);
- оплату праці фахівців, відповідальних за безпеку;
- навчання персоналу та аудит;
- втрати продуктивності через впровадження додаткових перевірок або політик.

У хмарних середовищах значну частку TCO становлять операційні витрати (OpEx), пов’язані з постійним використанням інструментів моніторингу. Для прикладу, активація повного аудиту AWS CloudTrail у кількох регіонах може збільшити місячний бюджет на 10–20 %, але суттєво знизити ризик недетектованих інцидентів.

Економічна доцільність контролів (Cost–Benefit Analysis)

Другим базовим показником є аналіз співвідношення витрат і вигод (Cost–Benefit Analysis, CBA), який дозволяє оцінити, чи виправдовує захід безпеки вкладені ресурси.

У найпростішому вигляді він може бути виражений як порівняння витрат на впровадження контролю ($C_{control}$) із очікуваними збитками від інциденту без цього контролю (L_{risk}):

$$\boldsymbol{C_{control} < L_{risk}}$$

Якщо витрати на контроль менші, ніж можливі втрати, захід вважається економічно обгрунтованим.

У більш практичних моделях (NIST SP 800-30, ISO/IEC 27005) оцінюється не лише ймовірність інциденту, а й залишковий ризик після впровадження заходів. Це дозволяє визначити граничну ефективність, коли подальше підвищення рівня безпеки не приносить пропорційної вигоди.

Показник рентабельності інвестицій у безпеку (Return on Security Investment)

Одним із найуживаніших економічних критеріїв є рентабельність інвестицій у безпеку (ROSI, Return on Security Investment).

Він показує, яку фінансову вигоду приносить впровадження певного засобу захисту у вигляді зменшення можливих втрат:

$$\boldsymbol{ROSI = \frac{(L_{risk} - L_{residual}) - C_{control}}{C_{control}} * 100\%}$$

де L_{risk} - очікувані збитки без заходів, $L_{residual}$ - залишкові збитки після їх упровадження.

Позитивне значення ROSI свідчить про економічну ефективність рішення, тоді як від'ємне - про перевитрати або неправильне розміщення пріоритетів.

Для хмарних середовищ цей показник є особливо корисним, адже дозволяє порівнювати різні підходи - наприклад, чи вигідніше використовувати вбудовані сервіси безпеки AWS (Security Hub, GuardDuty) чи впроваджувати зовнішні рішення типу Prisma Cloud.

Оцінка залишкового ризику (Residual Risk Index)

Ще одним важливим економічним критерієм є індекс залишкового ризику, який показує, який рівень ризику залишається після застосування засобів безпеки порівняно з допустимим рівнем.

Його можна визначити як:

$$\boldsymbol{Residual\ Risk\ Index = \frac{R_r}{R_0} * 100\%}$$

де R_0 - початковий ризик, R_r - ризик після впровадження контролів.

Цей показник відображає ефективність інвестицій у безпеку з погляду фактичного впливу на ризиковий профіль організації.

Якщо індекс перевищує встановлений поріг (наприклад, 30–40 %), доцільно переглянути політику інвестування у засоби захисту.

Вартість інцидентів та операційні втрати

Оцінюючи економічну ефективність, слід враховувати також вартість наслідків інцидентів, яка охоплює прямі фінансові збитки, витрати на розслідування, штрафи, втрату клієнтів і репутаційні ризики.

Згідно зі звітом IBM Cost of a Data Breach Report 2024, середня вартість порушення даних у хмарних середовищах становить приблизно 4,5 млн доларів США, причому компанії, які використовують автоматизовані системи безпеки, зменшують ці втрати в середньому на 40 %.

Це підтверджує, що економічно доцільним є саме інвестування в автоматизацію моніторингу та реагування, а не лише в розширення кількості контролів.

Загальна оптимізація витрат

Останнім аспектом є пошук оптимальної структури витрат на безпеку. Надмірна кількість інструментів, які дублюють функції, призводить до неефективного використання коштів.

Натомість інтегровані рішення (наприклад, Security Hub, Azure Defender або централізовані SIEM-платформи) забезпечують зменшення операційних витрат і кращу аналітичну узгодженість.

Оптимізація витрат також включає порівняння власних та керованих сервісів безпеки (Managed Security Services), що дозволяє компанії знайти баланс між контролем і вартістю підтримки.

Підсумовуючи, економічні критерії дозволяють перейти від інтуїтивного до раціонального управління безпекою, коли рішення приймаються на основі співвідношення витрат і вигод. Вони забезпечують стратегічну основу для вибору пріоритетів у захисті даних, дозволяють планувати бюджет і вимірювати віддачу від інвестицій.

У контексті хмарних середовищ економічна ефективність є не лише показником зрілості, а й умовою сталого функціонування - адже саме вона визначає, наскільки організація здатна підтримувати безпеку в довгостроковій перспективі.

2.3. Формалізація показників оцінювання

Розглянуті у попередніх підрозділах критерії - загальні, технічні, організаційні, часові та економічні - дають змогу комплексно оцінити стан системи безпеки, проте їх застосування потребує формалізації, тобто перетворення на чітко визначені показники, які можна виміряти, порівнювати та аналізувати у динаміці.

Формалізація показників є необхідною умовою для переходу від описового рівня оцінювання до кількісного, коли ефективність засобів захисту можна представити у вигляді конкретних числових значень або індексів. Це дозволяє:

- порівнювати результати між різними середовищами чи організаціями;
- виявляти тенденції у зміні рівня безпеки;
- автоматизувати моніторинг і звітність;
- будувати інтегральні моделі оцінювання (наприклад, Security Posture Index).

У нормативних документах NIST SP 800-55, ISO/IEC 27004 та CIS Controls Metrics підкреслюється, що процес формалізації повинен забезпечувати:

1. Вимірюваність (Measurability) - показник має бути виражений у кількісній формі;
2. Актуальність (Relevance) - показник повинен відображати суттєві аспекти безпеки;
3. Порівнянність (Comparability) - значення мають бути придатними для аналізу у часі;

4. Об’єктивність (Objectivity) - результати не повинні залежати від суб’єктивних оцінок.

Для хмарних середовищ формалізація має додаткове значення - вона створює основу для автоматизації. Сучасні платформи (AWS Security Hub, Azure Defender, GCP SCC, Prisma Cloud) використовують метрики у формалізованому вигляді, що дозволяє будувати узагальнені індекси безпеки для кожного акаунта, сервісу або організації.

У цьому підрозділі буде запропоновано систему метрик, які дозволяють кількісно оцінювати стан безпеки хмарного середовища, описано методику побудови інтегрального показника ефективності, а також принципи визначення вагових коефіцієнтів для різних груп критеріїв. Особливу увагу буде приділено питанням візуалізації та представлення результатів оцінювання, що є важливим для подальшої автоматизації у межах розробленої утиліти.

2.3.1. Побудова системи метрик для хмарного середовища

Побудова системи метрик є ключовим етапом формалізації оцінювання ефективності засобів захисту. Її метою є визначення набору кількісних показників, які точно відображають стан інформаційної безпеки у хмарній інфраструктурі та дозволяють контролювати зміни цього стану у часі.

На відміну від традиційних IT-систем, хмарні середовища характеризуються динамічністю, масштабованістю та великою кількістю взаємопов’язаних компонентів. Тому метрики мають не лише фіксувати фактичний стан безпеки, а й враховувати контекст середовища - тип сервісів (IaaS, PaaS, SaaS), модель відповідальності між провайдером і користувачем, а також рівень автоматизації контролів.

Принципи побудови системи метрик

Формування ефективної системи метрик базується на таких принципах:

1. Релевантність - метрика повинна мати безпосередній зв’язок із конкретною загрозою або процесом безпеки.
2. Уніфікованість - усі показники мають обчислюватися за єдиною методикою для забезпечення порівнянності результатів.
3. Масштабованість - система метрик має бути придатною для середовищ із різною кількістю акаунтів, регіонів і сервісів.
4. Автоматизованість збору даних - значення показників повинні отримуватись з об’єктивних джерел (API-виклики, журнали подій, сервіси моніторингу).
5. Інтерпретованість - результати повинні бути зрозумілими для користувачів різного рівня - від технічного персоналу до керівництва.

Класифікація метрик

Метрики доцільно поділяти за рівнем деталізації та функціональним призначенням на три групи:

1. Метрики стану (State Metrics) - описують поточний рівень безпеки системи.

Приклад: частка ресурсів, що відповідають політикам CIS AWS Foundations Benchmark, або кількість публічних S3-бакетів.

2. Метрики динаміки (Trend Metrics) - відображають зміни безпеки у часі, тобто наскільки система покращується або погіршується.

Приклад: зменшення кількості критичних уразливостей за квартал, тенденція скорочення середнього часу реагування (MTTR).

3. Метрики впливу (Impact Metrics) - оцінюють наслідки інцидентів або вплив заходів безпеки на бізнес-показники.

Приклад: зниження фінансових втрат від інцидентів, економія завдяки автоматизації перевірок безпеки.

Основні показники для хмарного середовища

Для практичної оцінки ефективності доцільно використовувати такі базові метрики, які можуть бути автоматично зібрані з API-інтерфейсів провайдерів:

Таблиця 4. Основні метрики для хмарного середовища

Категорія	Метрика	Формула або приклад
Відповідність	Compliance Rate	$\frac{N_{compliant}}{N_{total}} * 100\%$
Покриття	Coverage Index	$\frac{N_{controlled}}{N_{existing}} * 100\%$
Автоматизація	Automation Level	$\frac{N_{automated}}{N_{total}} * 100\%$
Реакція	MTTD / MTTR	середній час виявлення / реагування (у годинах)
Точність	Detection Accuracy	$\frac{TP}{TP + FP} * 100\%$
Усунення	Remediation Rate	$\frac{N_{fixed}}{N_{found}} * 100\%$
Стійкість	Availability (SLA)	частка часу, коли сервіс функціонує без збоїв
Економічна ефективність	ROSI	$\frac{(L_{risk} - L_{residual}) - C_{control}}{C_{control}} * 100\%$

Приклади реалізації у провайдерів

- У AWS Security Hub кожен акаунт отримує інтегральну оцінку Security Score, яка базується на відсотку виконаних перевірок за CIS, PCI DSS чи AWS Foundational Best Practices.

- У Azure Defender та Microsoft Secure Score аналогічна логіка використовується для підрахунку виконаних рекомендацій щодо налаштування мереж, ідентичностей та сховищ.
- Google Cloud Security Command Center (SCC) дозволяє будувати власні метрики на основі кількості активних вразливостей, статусів IAM-ролей і рівня шифрування даних.

Такі реалізації демонструють, що формалізовані показники не є лише теоретичними - вони використовуються у промислових рішеннях і можуть бути інтегровані у власні аналітичні системи або утиліти для аудиту безпеки.

Використання метрик у системі моніторингу

Після визначення набору метрик необхідно створити механізм їх збору, обчислення та візуалізації. Для цього можна застосовувати:

- збір даних через API (AWS SDK, boto3, Azure SDK, GCP Client Libraries);
- агрегацію результатів у централізованому сховищі (наприклад, DynamoDB, PostgreSQL або Elasticsearch);
- візуалізацію у вигляді дашбордів (Grafana, CloudWatch Dashboards, Kibana).

Формалізована система метрик дозволяє проводити не лише оцінку поточного стану, але й виявляти тенденції, прогнозувати ризики та планувати подальші заходи. Зокрема, у межах цієї роботи такі метрики будуть використані як основа для створення утиліти, що аналізує безпеку AWS-акаунта та формує звіт про відповідність політикам безпеки.

2.3.2. Кількісна шкала оцінювання: оцінка відхилень від базового рівня безпеки

Після визначення системи метрик наступним етапом є побудова кількісної шкали оцінювання, яка дозволяє порівнювати значення показників між різними компонентами, сервісами чи часовими періодами. Метою цього процесу є не лише фіксація абсолютних значень, а й визначення ступеня відхилення від цільового або базового рівня безпеки.

Поняття базового рівня безпеки

Базовий рівень безпеки (Baseline Security Level) визначається як еталонний стан системи, що відповідає мінімальним вимогам до конфігурації та політик безпеки.

У міжнародній практиці такий рівень встановлюється відповідно до:

- CIS Benchmarks - контрольних вимог до налаштування сервісів (наприклад, AWS CIS Level 1/2);
- NIST SP 800-53 або ISO/IEC 27017 - наборів технічних і адміністративних контролів;
- внутрішніх стандартів компанії, затверджених службою інформаційної безпеки.

Базовий рівень використовується як точка відліку, від якої вимірюється фактичний стан безпеки. У хмарному середовищі це може бути набір перевірок, що автоматично виконуються засобами AWS Security Hub або інструментами policy-as-code (наприклад, Checkov, tfsec, OPA).

Шкала оцінювання та нормалізація показників

Оскільки метрики мають різні одиниці вимірювання (відсотки, години, кількість інцидентів тощо), для побудови інтегральної оцінки необхідно привести їх до єдиної нормалізованої шкали, зазвичай у діапазоні [0; 1] або [0; 100 %].

Загальна формула нормалізації має вигляд:

$$\mathbf{N_i = \frac{X_i - X_{min}}{X_{max} - X_{min}}}$$

де N_i - нормалізоване значення метрики, X_i - фактичне значення, X_{min} та X_{max} - межі прийнятних значень.

Для показників, де менше - краще (наприклад, час реагування або кількість інцидентів), формула набуває вигляду:

$$\mathbf{N_i = 1 - \frac{X_i - X_{min}}{X_{max} - X_{min}}}$$

Таким чином, усі метрики можна подати на єдиній шкалі, де 1 (або 100 %) означає повну відповідність базовому рівню, а 0 - критичне відхилення.

Інтерпретація шкали

Для зручності аналізу пропонується розділити шкалу оцінювання на чотири рівні, які відображають стан системи:

Таблиця 5. Шкала для визначення допустимий рівень безпеки середовища

Рівень безпеки	Діапазон значень	Інтерпретація
Високий (Задовільний)	0.85 – 1.00	Система відповідає вимогам, ризики мінімальні
Середній (Прийнятний)	0.70 – 0.84	Існують незначні відхилення, але контроль ефективний
Низький (Підвищений ризик)	0.50 – 0.69	Значна кількість недоліків, потрібні коригувальні дії
Критичний (Незадовільний)	< 0.50	Система перебуває у стані підвищеної загрози

Ця градація дозволяє оперативно класифікувати стан середовища та встановити пороги сповіщень або автоматичних дій у разі перевищення критичних меж.

Оцінка відхилень у динаміці

Для моніторингу ефективності у часі важливо відстежувати відхилення показників від базового рівня. Таке відхилення може бути представлене у вигляді приросту або зниження інтегрального показника:

$$\Delta \mathbf{N_i} = \mathbf{N_i}(\mathbf{t_2}) - \mathbf{N_i}(\mathbf{t_1})$$

де t_1 та t_2 - початковий і поточний періоди спостереження.

Позитивне значення ΔN_i означає покращення рівня безпеки (зменшення відхилення), а негативне - погіршення стану.

На практиці такий підхід використовується у AWS Security Hub і Microsoft Secure Score, де інтерфейс відображає не лише поточний рівень безпеки, а й динаміку його змін протягом певного періоду.

Використання шкали для порівняльного аналізу

У багатохмарних або мультиакаунтних середовищах кількісна шкала дозволяє:

- порівнювати безпеку між різними акаунтами, командами чи регіонами;
- визначати пріоритетні зони для покращення;
- оцінювати вплив нових політик чи технічних заходів на загальний рівень безпеки.

Для прикладу, якщо середній Compliance Score у європейських акаунтах становить 0.91, а в азійських - 0.67, це сигналізує про необхідність уніфікації політик або додаткового контролю в певному регіоні.

Таким чином, кількісна шкала оцінювання є основою аналітичного підходу до вимірювання ефективності безпеки. Вона перетворює розрізнені метрики на уніфіковану систему, придатну для порівняння, автоматичного контролю та побудови інтегральних індексів.

2.3.3. Формули для розрахунку інтегрального показника

Коли окремі показники безпеки (відповідність, автоматизація, точність, своєчасність тощо) уже визначено і нормалізовано, постає завдання отримати узагальнену числову оцінку, яка відображає загальний рівень захищеності системи. Такий інтегральний показник називають індексом ефективності безпеки або Security Posture Score (SPS).

Концепція інтегрального показника

Ідея полягає у тому, щоб об’єднати всі нормалізовані метрики в єдину формулу з урахуванням їхньої вагомості - ступеня впливу на загальний рівень безпеки.

Таким чином, кожен критерій (наприклад, відповідність чи своєчасність) робить свій внесок у підсумковий бал пропорційно до його важливості.

У спрощеній формі інтегральний показник можна подати як зважене середнє:

$$\mathbf{SPS} = \sum_{\mathbf{i=1}}^{\mathbf{n}} \boldsymbol{\omega}_\mathbf{i} * \mathbf{N}_\mathbf{i}$$

де SPS - інтегральний показник ефективності безпеки;
N_i - нормалізоване значення і-го показника (у діапазоні [0;1]);
$\boldsymbol{\omega}_\mathbf{i}$ - ваговий коефіцієнт для і-го показника (сума всіх $\boldsymbol{\omega}_\mathbf{i}$= 1);
n - кількість показників у моделі.

Визначення вагових коефіцієнтів

Вагові коефіцієнти визначаються залежно від пріоритетів організації або контексту середовища.
Наприклад, у фінансових установах вищу вагу мають показники відповідності (compliance) та цілісності даних, тоді як у DevOps-орієнтованих компаніях більшу роль відіграють автоматизація та швидкість реагування.

У практиці управління безпекою ваги можна встановлювати за допомогою таких методів:

- Експертного оцінювання (Expert Scoring) - група фахівців визначає важливість кожного критерію в межах 0–1;
- Методу аналітичної ієрархії (AHP) - ваги розраховуються шляхом попарного порівняння критеріїв;
- Емпіричного підходу - ваги визначаються на основі статистики інцидентів або впливу ризиків.

Для хмарного середовища доцільно застосовувати комбінований підхід, де технічні, організаційні, часові та економічні критерії мають пропорційний вплив. Приклад розподілу ваг наведено нижче:

Таблиця 6. Приклад вагових показників

Група критеріїв	Прикладні показники	Вага ($\boldsymbol{\omega}_\mathbf{i}$)
Технічні	Автоматизація, точність, покриття	0.35
Організаційні	Зрілість процесів, DevSecOps, обізнаність	0.25
Часові	MTTD, MTTR, SLA	0.20
Економічні	ROSI, TCO, залишковий ризик	0.15
Інші (контекстні)	Специфічні KPI компанії	0.05

Приклад розрахунку

Для прикладу розглянемо умовний набір нормалізованих показників хмарної системи:

Таблиця 7. Приклад значеннь нормалізованих параметрів і вагових показників

Критерій	Позначення	Значення N_i	Вага $\boldsymbol{\omega_i}$
Compliance Rate	N_1	0.92	0.20
Automation Level	N_2	0.85	0.15
Detection Accuracy	N_3	0.88	0.10
MTTD / MTTR	N_4	0.75	0.15
Security Maturity	N_5	0.80	0.15
ROSI	N_6	0.70	0.10
Consistency	N_7	0.95	0.15

Тоді інтегральний показник буде:

$$\mathbf{SPS} = (\mathbf{0.92} * \mathbf{0.20}) + (\mathbf{0.85} * \mathbf{0.15}) + (\mathbf{0.88} * \mathbf{0.10}) + (\mathbf{0.75} * \mathbf{0.15}) + (\mathbf{0.70} * \mathbf{0.10}) + (\mathbf{0.95} * \mathbf{0.15})$$

$$\mathbf{SPS} = \mathbf{0.83}$$

Отже, загальний рівень безпеки становить 0.83, що відповідає середнь рівню (прийнятний стан із незначними відхиленнями).

Такий підхід дозволяє отримати об’єктивну й порівняльну оцінку стану безпеки, яку можна застосовувати:

- для внутрішнього моніторингу у межах компанії;
- для звітності перед аудиторами або керівництвом;
- для автоматичного визначення пріоритетів у планах усунення вразливостей.

Практична реалізація у хмарному моніторингу

Більшість сучасних інструментів CSPM використовують подібні моделі для розрахунку узагальнених індексів безпеки. Наприклад:

- AWS Security Hub обчислює Security Score як середнє виконання контрольних правил з урахуванням їхньої ваги.
- Azure Secure Score використовує аналогічну логіку, присвоюючи більшу вагу контролям із високим рівнем ризику.
- Google Security Command Center (SCC) надає Risk Index, де вплив кожного порушення оцінюється за формулою “ймовірність × наслідок”.

Таким чином, інтегральний показник є не лише аналітичним інструментом, а й практичним механізмом управління ризиками: він дозволяє пріоритизувати заходи, відстежувати ефективність політик і наочно демонструвати динаміку поліпшення безпеки.

Варіації моделі залежно від типу організації

Залежно від характеру діяльності та зрілості організації ваги можуть суттєво відрізнятися.

У таблиці нижче наведено орієнтовні приклади трьох типових моделей.

Таблиця 8. Приклади вагових показників для компаній різних типів

Тип організації	Технічні	Організаційні	Часові	Економічні	Коментар
IT-компанія / стартап	0.45	0.20	0.25	0.10	Основний акцент на автоматизації, CI/CD, швидкості реагування; менше уваги до комплаєнсу
Фінансова або банківська установа	0.30	0.30	0.15	0.25	Висока цінність даних → підвищена вага організаційних і економічних критеріїв
Державна установа / критична інфраструктура	0.25	0.35	0.25	0.15	Пріоритет на нормативній відповідності, управлінні процесами та стабільності

Адаптація коефіцієнтів у мультихмарному середовищі

У мультихмарних системах (AWS + Azure + GCP) часто спостерігається нерівномірність рівнів безпеки між платформами. Тому модель ваг має враховувати:

- рівень автоматизації кожного провайдера (наприклад, Azure Policy має більше вбудованих засобів відповідності, ніж GCP Config Connector);
- наявність спільних сервісів моніторингу;
- співвідношення між контрольованими і неконтрольованими зонами (наприклад, SaaS-компоненти).

Для таких випадків доцільно вводити адаптивні ваги, що змінюються залежно від ступеня охоплення сервісів:

$$\omega_i' = \omega_i * (1 + \alpha_i)$$

де α_i - коефіцієнт корекції для даної платформи або регіону.

Використання ваг у подальшому аналізі

Після встановлення вагових коефіцієнтів інтегральний показник (SPS) може використовуватись не лише для оцінки поточного стану, а й для порівняльного аналізу між середовищами, побудови пріоритетів виправлення або оцінки ефективності нових політик.

Наприклад:
- підвищення ваги часових критеріїв після інциденту дозволяє оперативно оцінити успішність заходів із покращення швидкості реагування;
- зниження ваги економічних показників у період активного впровадження DevSecOps допомагає уникнути перекосів у короткострокових оцінках.

Використання ваг у подальшому аналізі

Після встановлення вагових коефіцієнтів інтегральний показник (SPS) може використовуватись не лише для оцінки поточного стану, а й для порівняльного аналізу між середовищами, побудови пріоритетів виправлення або оцінки ефективності нових політик.

Наприклад:
- підвищення ваги часових критеріїв після інциденту дозволяє оперативно оцінити успішність заходів із покращення швидкості реагування;
- зниження ваги економічних показників у період активного впровадження DevSecOps допомагає уникнути перекосів у короткострокових оцінках.

Таким чином, модель вагових коефіцієнтів є гнучким механізмом адаптації оцінки ефективності до контексту організації. Вона забезпечує баланс між технічними, організаційними та економічними аспектами безпеки, дозволяючи отримувати релевантні результати як для стратегічного планування, так і для оперативного моніторингу стану захищеності системи.

2.3.4. Приклади оцінювання існуючих систем

Для перевірки практичної придатності запропонованих критеріїв та метрик було здійснено порівняльний аналіз кількох сучасних рішень у сфері хмарної безпеки. Основна мета полягала у тому, щоб визначити, наскільки ефективно різні системи реалізують ключові функції захисту, та як їхні можливості співвідносяться з визначеними критеріями ефективності.

Об’єктами оцінювання обрано три найбільш поширені платформи у своїх екосистемах:
1. AWS Security Hub - централізований сервіс моніторингу стану безпеки в Amazon Web Services.
2. Microsoft Defender for Cloud (раніше Azure Security Center) - інтегрована платформа захисту ресурсів у середовищі Microsoft Azure.

3. Prisma Cloud (Palo Alto Networks) - мультихмарне рішення класу Cloud Security Posture Management (CSPM), орієнтоване на комплексну аналітику та автоматизацію.

AWS Security Hub

Сервіс AWS Security Hub є складовою екосистеми Amazon і забезпечує збір та аналіз подій безпеки з різних джерел - GuardDuty, Config, Inspector, Macie, CloudTrail.

Основною особливістю є агрегація результатів перевірок відповідно до стандартів безпеки (CIS AWS Foundations, PCI DSS, NIST CSF) та розрахунок інтегрального Security Score.

Оцінка за критеріями:

- Відповідність: високий рівень (понад 90 % покриття основних контрольних пунктів CIS і NIST).
- Повнота: система охоплює більшість сервісів AWS, але не аналізує зовнішні інтеграції чи сторонні сервіси.
- Автоматизація: високий рівень - є інтеграція з Lambda для автоусунення порушень.
- Своєчасність: забезпечується у режимі майже реального часу, затримка мінімальна.
- Економічна ефективність: сервіс вбудований у AWS і не потребує складної підтримки, однак витрати зростають зі збільшенням обсягу журналів (CloudTrail, Config).

Перевагою AWS Security Hub є тісна інтеграція з нативними сервісами та можливість побудови автоматизованих сценаріїв реагування. Недоліком - обмеженість у мультихмарних сценаріях: платформа охоплює лише ресурси AWS.

Microsoft Defender for Cloud

Microsoft Defender for Cloud - це інтегрована система безпеки Azure, яка об’єднує функції моніторингу, управління уразливостями, виявлення загроз і рекомендацій щодо усунення ризиків.

Платформа оцінює безпеку на основі контрольних профілів (наприклад, ISO 27001, CIS, NIST SP 800-53) та формує Secure Score для кожного підписника.

Оцінка за критеріями:

- Відповідність: високий рівень завдяки підтримці понад 20 стандартів і можливості додавання власних політик.
- Повнота: охоплює не лише хмарні, а й локальні ресурси через Azure Arc, що розширює застосування.
- Автоматизація: частково реалізована - певні дії виконуються вручну, хоча підтримується інтеграція з Logic Apps і Sentinel.

- Своєчасність: висока, але є затримки в оновленні статусів через періодичність сканувань.
- Економічна ефективність: гнучка модель оплати, проте витрати можуть суттєво зрости при великій кількості підписок.

Основною перевагою Defender for Cloud є універсальність і глибока інтеграція з інфраструктурою Azure, включно з Microsoft Sentinel, що спрощує реалізацію принципу “end-to-end security”. Недолік - менша прозорість у кастомізації алгоритмів оцінювання, що ускладнює точне налаштування метрик.
Prisma Cloud (Palo Alto Networks)

Prisma Cloud є одним із найповніших комерційних рішень класу CSPM / CNAPP, яке підтримує всі основні хмарні платформи (AWS, Azure, GCP, Oracle Cloud) і контейнери Kubernetes.

Вона поєднує в собі функції моніторингу конфігурацій, керування вразливостями, аналізу IAM-політик та забезпечення комплаєнсу.

Оцінка за критеріями:

- Відповідність: дуже високий рівень - понад 20 шаблонів стандартів безпеки, включно з ISO, NIST, PCI DSS, GDPR.
- Повнота: максимальна серед розглянутих - підтримка мультихмарності, контейнерів, безсерверних функцій.
- Автоматизація: наявна, з розвиненими функціями автоусунення, інтеграцією з Terraform та API-орієнтованим управлінням.
- Своєчасність: залежить від налаштування частоти сканувань, але при постійній інтеграції працює майже в реальному часі.
- Економічна ефективність: висока ціна, однак виправдана для середніх і великих організацій завдяки широким можливостям централізованого контролю.

Prisma Cloud демонструє найвищий рівень гнучкості та глибини аналізу, проте її використання економічно виправдане лише для масштабних компаній або мультихмарних середовищ.

2.3.5. Аналіз сильних і слабких сторін рішень відповідно до критеріїв

Порівняння сучасних систем хмарної безпеки за технічними, організаційними, часовими та економічними критеріями дає змогу глибше оцінити їхню придатність у різних типах середовищ. Кожне з рішень має власну архітектурну філософію та набір функцій, що формують його конкурентні переваги й обмеження. Нижче наведено аналітичний огляд сильних і слабких сторін AWS Security Hub, Microsoft Defender for Cloud та Prisma Cloud відповідно до раніше визначених критеріїв.

AWS Security Hub
Сильні сторони:

- Глибока інтеграція у власну екосистему. Сервіс безпосередньо взаємодіє з GuardDuty, Macie, Inspector, CloudTrail, що забезпечує значно точнішу кореляцію подій, ніж сторонні рішення.
- Високий рівень автоматизації. У поєднанні з AWS Lambda та EventBridge дозволяє будувати ефективні механізми auto-remediation.
- Простота впровадження. Security Hub не потребує розгортання агентів або окремої інфраструктури - достатньо увімкнути сервіс.
- Стандартизована система оцінювання. Security Score базується на добре відомих CIS Benchmarks та AWS Foundational Best Practices, що спрощує аудити.

Слабкі сторони:
- Відсутність мультихмарності. Security Hub не призначений для контролю середовищ за межами AWS, що є суттєвим обмеженням у великих компаніях.
- Обмеженість аналітики IAM. Хоч сервіс показує рекомендації, глибокий аналіз політик ролей (наприклад, надлишкових дозволів) потребує додаткових рішень.
- Вартість при масштабуванні. Великі обсяги логів CloudTrail і Config можуть істотно збільшувати операційні витрати.

Microsoft Defender for Cloud
Сильні сторони:
- Гібридне покриття завдяки Azure Arc. Дозволяє підключати локальні сервери, Kubernetes-кластери та інші середовища, що робить його корисним для організацій, які працюють у гібридній моделі.
- Широка інтеграція в Microsoft Stack. Зв’язка із Sentinel, Purview, 365 Defender формує єдину лінію оборони.
- Сильні можливості з управління комплаєнсом. Підтримує велику кількість стандартів і надає чітко структуровані рекомендації.
- Хороший аналіз конфігурацій мережі. Defender for Cloud має розвинені інструменти для аналізу мережевих залежностей у Azure.

Слабкі сторони:
- Відносно менша гнучкість автоматизації. Хоча Logic Apps потужні, автоусунення не так легко стандартизувати, як у AWS.
- Затримки в оновленні стану. Деякі перевірки виконуються періодично, тому статуси ризиків не завжди в режимі реального часу.
- Складність конфігурації. Велика кількість модулів і рівнів призводить до високої кривої навчання.

Prisma Cloud (Palo Alto Networks)
Сильні сторони:
- Універсальність і мультихмарність. Підтримує AWS, Azure, GCP, OCI, Kubernetes, serverless, а також IaC. Це робить його майже унікальним серед CSPM/CNAPP-рішень.

- Глибокий IAM-аналітик. Аналізує надлишкові дозволи, виявляє ризикові ролі та пропонує оптимізацію.
- Потужні можливості CI/CD-сканування. Інтеграція з GitHub, GitLab, Jenkins, ArgoCD забезпечує перевірку IaC ще до деплою.
- Розвинені механізми auto-remediation та policy-as-code. Це підсилює DevSecOps-підхід і мінімізує людський фактор.
- Висока деталізація комплаєнсу. Платформа підтримує десятки стандартів, включно з фінансовими й державними.

Слабкі сторони:

- Висока вартість володіння. Підписка є дорогою, що робить продукт менш доступним для малих компаній.
- Складність впровадження. Повноцінний запуск потребує налаштування ролей, інтеграцій, агентів та політик.
- Високий поріг компетенцій. Необхідні фахівці зі знанням багатьох платформ і DevSecOps-практик.

Таблиця 9. Порівняльний аналіз відповідно до критеріїв

Критерій	Найсильніший	Найслабший	Коментар
Покриття	Prisma Cloud	AWS Security Hub	Prisma охоплює мультихмару та контейнери; Security Hub - тільки AWS
Автоматизація	AWS Security Hub / Prisma Cloud	Defender for Cloud	AWS найкраще інтегрує автоусунення; Prisma - гнучкіший, але складніший
Організаційна інтеграція	Defender for Cloud	Security Hub	Microsoft повністю побудував єдиний security-stack
IAM-аналітика	Prisma Cloud	AWS Security Hub	Prisma аналізує надлишкові дозволи глибше
Своєчасність	AWS Security Hub	Defender for Cloud	AWS працює майже у реальному часі
Економічна ефективність	Security Hub (для AWS-компаній)	Prisma Cloud	Security Hub дешевший у власній екосистемі
Універсальність	Prisma Cloud	Security Hub	Prisma для мультихмари; Security Hub - vendor-locked

Загальні висновки щодо сильних і слабких сторін

Аналіз показує, що жодне з рішень не є універсальним для всіх сценаріїв.

- AWS Security Hub оптимальний для організацій, що повністю використовують AWS й прагнуть максимальної автоматизації без надлишкових витрат.
- Microsoft Defender for Cloud найкраще підходить середовищам із гібридною інфраструктурою та глибокою інтеграцією Microsoft-рішень.
- Prisma Cloud є найбільш масштабованим та універсальним варіантом для мультихмарних середовищ і DevSecOps-орієнтованих команд, однак потребує значних ресурсів для впровадження та підтримки.

Висновки до розділу 2

Другий розділ був присвячений формуванню системи критеріїв та показників, необхідних для об’єктивної оцінки ефективності засобів захисту інформації у хмарному середовищі. Завданням розділу було не лише визначити перелік критеріїв, а й побудувати методологічну основу, яка дозволяє перетворити якісні характеристики безпеки на кількісні метрики, придатні для порівняння, моніторингу та автоматизації.

У першій частині розділу було проаналізовано, як у сучасних стандартах (ISO/IEC 27004, NIST SP 800-55, CIS Benchmarks) та наукових дослідженнях інтерпретується поняття ефективності. Особливо підкреслено, що у сфері хмарної безпеки ефективність не може оцінюватися лише за технічними параметрами - необхідно враховувати організаційні процеси, швидкість реагування на інциденти, відповідність нормативам і економічну доцільність заходів. Такий широкий підхід дозволив сформувати комплексну систему критеріїв, яка охоплює технічні, часові, організаційні та економічні аспекти.

Одним із ключових результатів розділу стало формування системи метрик, що дозволяє кількісно вимірювати стан безпеки. Це особливо важливо у хмарному середовищі, де швидкість змін, висока динаміка ресурсів та автоматизація процесів унеможливлюють традиційні “ручні” підходи до оцінювання. Метрики відповідності, покриття, точності виявлення, автоматизації, часу виявлення (MTTD), часу реагування (MTTR), рівня залишкового ризику та економічної віддачі (ROSI) дають змогу оцінити захисні механізми у вимірюваній формі.

Особливу увагу було приділено нормалізації показників, оскільки різні метрики мають різну природу: одні вимірюються у відсотках, інші - у годинах чи кількості зафіксованих інцидентів. Нормалізована шкала у діапазоні [0;1] дозволяє усунути різнорідність даних і формувати єдині узагальнені індекси. Це робить можливим об’єктивне порівняння різних сервісів або акаунтів, а також побудову рівнів ризику.

Важливою складовою стало обгрунтування інтегрального показника ефективності - Security Posture Score (SPS). На відміну від одиничних метрик, SPS дозволяє оцінювати стан безпеки комплексно, враховуючи не лише виконання окремих контролів, а й їхню вагу. Підхід із використанням вагових коефіцієнтів відображає реальний контекст організації: у фінансових установах

пріоритетними є комплаєнс і неприйнятність ризику, тоді як технологічні компанії акцентують увагу на швидкості реагування та автоматизації.

Окремий результат - обгрунтування моделей визначення вагових коефіцієнтів. Аналіз показав, що ваги не можуть бути універсальними: вони повинні змінюватися залежно від типу бізнесу, архітектури хмарного середовища, рівня зрілості ІБ-процесів і навіть специфіки командної культури. Було запропоновано декілька методів визначення ваг - експертний, метод аналітичної ієрархії (AHP), адаптивний та статистичний, - кожен із яких має свої переваги та обмеження. Це забезпечує гнучкість у побудові моделі SPS і дозволяє адаптувати її до вимог конкретної організації.

Друга частина розділу була орієнтована на практичне застосування розробленої методології. На прикладі трьох сучасних рішень - AWS Security Hub, Microsoft Defender for Cloud та Prisma Cloud - продемонстровано, як критерії і метрики можуть оцінюватися у реальних умовах. Порівняння рівнів покриття, автоматизації, інтеграції, економічної доцільності та аналітичних можливостей показало, що сильні й слабкі сторони рішень суттєво впливають на їхній інтегральний показник SPS. Наприклад, Prisma Cloud отримує високі оцінки за рахунок мультихмарності та глибокого IAM-аналізу, тоді як AWS Security Hub сильний у автоматизації нативних сервісів, але обмежений у мультиплатформеності.

Загальний висновок розділу полягає в тому, що ефективність засобів безпеки повинна оцінюватися в багатовимірному просторі, а не за окремими показниками. Хмарна специфіка - масштабованість, динамічне створення ресурсів, широке використання DevOps-підходів - потребує чітких, кількісно визначених, автоматизованих метрик. Лише така система дозволяє виявляти тенденції, прогнозувати ризики, порівнювати платформи та приймати обгрунтовані рішення щодо оптимізації заходів безпеки.

Розроблена в цьому розділі методологія оцінювання є фундаментом для подальших рекомендацій та реалізації програмної утиліти, яка буде аналізувати хмарне середовище та надавати автоматизовані звіти. Саме на цій основі будується логіка наступного розділу, у якому будуть запропоновані конкретні підходи до підвищення рівня безпеки та методи їх практичної реалізації.

Розділ 3 Розробка рекомендацій та методів підвищення ефективності захисту інформації в хмарному середовищі

Результати аналізу, проведеного у попередніх розділах, показали, що ефективність системи інформаційної безпеки у хмарному середовищі залежить від низки взаємопов’язаних чинників: якості технічних контролів, зрілості організаційних процесів, рівня автоматизації, своєчасності реагування на інциденти та економічної обгрунтованості впроваджених заходів. Крім того, аналіз сучасних хмарних рішень продемонстрував, що жодне з них не забезпечує повного покриття всіх аспектів безпеки, і тому організації змушені формувати власні підходи до підвищення рівня захищеності.

У цьому контексті виникає потреба у формуванні системи рекомендацій, яка, грунтуючись на визначених критеріях та формалізованих показниках ефективності, дозволить цілеспрямовано покращувати стан безпеки у хмарній інфраструктурі. Такі рекомендації мають не лише враховувати специфіку хмарних платформ, але й бути адаптованими до реальних умов - різного рівня зрілості процесів, масштабів інфраструктури, вимог до комплаєнсу та стратегічних цілей організації.

Третій розділ спрямований на розробку практично застосовних методів удосконалення безпеки, які можуть бути реалізовані як безпосередньо у хмарному середовищі, так і на рівні процесів управління. Особливу увагу приділено рекомендаціям, здатним підвищити ключові метрики, визначені у другому розділі, - такі як рівень відповідності, швидкість реагування, точність виявлення порушень та інтегральний показник ефективності (SPS). На основі запропонованої методології створено приклад програмної утиліти для аналізу безпеки AWS-акаунта, що демонструє можливості автоматизації оцінювання та генерування рекомендацій.

Таким чином, цей розділ поєднує теоретичні засади з практичними інструментами та формує основу для подальших висновків роботи. Він містить як загальні принципи підвищення безпеки, так і конкретні технічні, організаційні та економічні рекомендації, організовані відповідно до розробленої системи критеріїв і метричних моделей.

3.1. Загальні принципи підвищення рівня безпеки у хмарних середовищах

3.1.1. Принцип найменших привілеїв у розподілених системах

Одним із базових принципів побудови захищених інформаційних систем, особливо у хмарному середовищі, є принцип найменших привілеїв (Principle of Least Privilege, PoLP). Його суть полягає в тому, що кожному користувачу, сервісу або компоненту надається мінімально необхідний обсяг прав, достатній лише для виконання покладених на нього функцій. У розподілених і динамічних архітектурах - таких як хмарні платформи - цей принцип є не просто “хорошою практикою”, а критично важливим механізмом обмеження потенційних наслідків помилок та компрометації облікових даних.

У традиційних IT-середовищах надмірні привілеї часто залишаються локальною проблемою окремих систем. У хмарі ситуація інша: один невдало налаштований обліковий запис або роль із широкими правами може мати доступ до значної частини інфраструктури - баз даних, сховищ, журналів, систем управління ключами. Це означає, що порушення принципу найменших привілеїв прямо впливає на масштаб можливого інциденту: чим ширші права, тим більший потенційний збиток.

З практичної точки зору реалізація цього принципу в хмарному середовищі тісно пов’язана з механізмами керування ідентичностями та доступом (IAM) та рольовими моделями. У більшості провайдерів (AWS IAM, Azure RBAC, Google Cloud IAM) доступ до ресурсів визначається через політики, ролі та прив’язку цих ролей до суб’єктів (користувачів, груп, сервісних акаунтів, функцій). Відповідно, від якості проєктування цих політик залежить, чи буде принцип PoLP дійсно дотримуватись, чи залишиться лише декларацією в документації.

Одним із ключових організаційних викликів є те, що в реальних умовах правила доступу мають тенденцію “розширюватися”. Щоб швидко вирішити операційну задачу, адміністратори або розробники часто додають до політики зайві дозволи (“щоб точно спрацювало”) і пізніше не повертаються до їхнього перегляду. З часом це призводить до накопичення великої кількості ролей із надмірними правами, що суттєво знижує загальний рівень безпеки.

Для хмарної інфраструктури важливо не лише декларувати принцип найменших привілеїв, а й забезпечити процесний і технічний контроль його дотримання. Це включає:

- Чітке визначення ролей і зон відповідальності. Кожна роль (як технічна, так і організаційна) має відповідати конкретним функціям: адміністратор БД, DevOps-інженер, аналітик безпеки, сервісна функція тощо.
- Використання рольового доступу замість прямих прав. Замість того щоб призначати привілеї окремим акаунтам, доступ делегується через ролі, які легше контролювати, переглядати та змінювати.
- Регулярний перегляд та “очищення” політик. Потрібно періодично аналізувати, які дозволи фактично використовуються, і вилучати зайві. Для цього можуть застосовуватися як вбудовані засоби провайдерів, так і зовнішні інструменти аналізу IAM.
- Розмежування робочих середовищ. Відокремлення доступів до dev, test і production-середовищ зменшує ризики неконтрольованих змін у критичних системах.

У контексті метрик, розроблених у другому розділі, реалізація принципу найменших привілеїв безпосередньо впливає на:

- зниження залишкового ризику,
- покращення показників відповідності (Compliance Rate) до внутрішніх політик та стандартів,
- зменшення ймовірності інцидентів, пов’язаних із компрометацією доступу,

- а також спрощення аналізу інцидентів, оскільки дії кожної ролі чітко обмежені.

Важливо, що PoLP є не лише технічним, а й культурним принципом. Він вимагає від організації відмови від підходу “дешевше дати більше доступу, ніж розбиратися”, натомість впроваджуючи практику “спочатку мінімально необхідне - потім, за обгрунтуванням, додаткове”. Для хмарних середовищ, де все більше операцій відбувається автоматизовано (CI/CD, інфраструктура як код, автоматичні деплої), такий підхід стає невід’ємною складовою загальної архітектури безпеки.

3.1.2. Принцип багаторівневого захисту (Defense-in-Depth)

Принцип багаторівневого захисту, або Defense-in-Depth, є однією з найстаріших, але й сьогодні - однією з найактуальніших концепцій побудови інформаційної безпеки. Його основна ідея полягає в тому, що жоден окремий захисний механізм не може гарантувати повну безпеку; натомість ефективність досягається завдяки поєднанню кількох рівнів контролю, які взаємно доповнюють один одного і забезпечують стійкість системи навіть у разі відмови або обходу одного із них.

У контексті хмарних середовищ принцип багаторівневого захисту набуває особливої ваги. Хмара не є лінійною або статичною архітектурою - це динамічне, розподілене середовище з різними рівнями абстракції: інфраструктура, мережа, обчислювальні ресурси, платформи, дані, ідентичності. Кожен рівень має власні ризики, які не завжди можливо усунути одним засобом. Тому саме комбіновані та послідовні захисні механізми створюють реальну стійкість.

У класичному вигляді модель Defense-in-Depth передбачає наявність таких рівнів:

- мережевий рівень (ізоляція підмереж, firewall-контролі, контроль трафіку);
- рівень доступу та ідентифікацій (IAM, MFA, контроль привілеїв);
- рівень застосунків та сервісів (вразливості, виправлення, контроль конфігурацій);
- рівень даних (шифрування, контроль доступу, резервування);
- рівень моніторингу та реагування (логування, аналіз подій, автоматизовані дії);
- процесний рівень (політики, організаційні процедури, аудит).

У хмарних платформах ці рівні зберігаються, але доповнюються специфічними аспектами: ізоляцією робочих середовищ, безсерверними архітектурами, контейнерними оркестраторами, сервісами автоматичного масштабування, доступом через API тощо. Відповідно, багато з традиційних ризиків набувають нової форми та вимагають оновлених методів стримування.

Практичне значення Defense-in-Depth полягає в тому, що навіть якщо порушнику вдалося подолати один із рубежів захисту, інші шари продовжують виконувати стримувальну функцію. Наприклад, компрометація однієї IAM-ролі

не повинна автоматично дозволяти зловмиснику отримати доступ до критичних даних, якщо:

- на мережевому рівні діє сегментація,
- S3 або інші сховища додатково захищені політиками доступу,
- логування забезпечує швидке виявлення аномалій,
- дані зашифровані окремими ключами KMS,
- застосунки виконують додаткову перевірку запитів.

По суті, багаторівневість перетворює систему на мозаїку взаємопов'язаних бар'єрів, де кожен окремо може бути неповним, але разом вони істотно ускладнюють реалізацію атаки.

З точки зору оцінювання ефективності (показники, розроблені у розділі 2), Defense-in-Depth безпосередньо впливає на:

- Coverage Index - чим більше шарів охоплено політиками, тим повніший контроль системи;
- Residual Risk - комбіновані бар'єри дозволяють знизити залишковий ризик до допустимого рівня;
- Detection Accuracy - розподілене логування та кореляція подій покращують точність виявлення інцидентів;
- MTTD/MTTR - багаторівневий моніторинг скорочує час виявлення і реагування;
- Compliance Rate - більшість стандартів безпеки (ISO 27017, CIS, NIST CSF) прямо вимагає реалізації багаторівневого підходу.

Окремо важливо, що принцип Defense-in-Depth добре поєднується з архітектурними моделями сучасних хмар, зокрема з підходами Zero Trust, segmentation, least privilege та DevSecOps, утворюючи комплексну стратегічну рамку безпеки.

З погляду організації, багаторівневий захист також сприяє підвищенню стійкості до людського фактору. Навіть якщо адміністративна помилка або некоректне налаштування призводять до виникнення вразливості на одному рівні, інші рівні все ще здатні компенсувати ризик і захистити критичні активи.

3.1.3. Автоматизація як основа сучасної хмарної безпеки

Автоматизація стала одним із ключових принципів сучасної хмарної безпеки, і це не випадково. Статичні або ручні моделі контролю, які колись були достатньо ефективними у традиційних дата-центрах, у хмарних середовищах втрачають свою актуальність. Хмара - це динамічна екосистема, де ресурси створюються, видаляються та змінюються за секунди, а значна частина дій відбувається без прямої участі людини. У таких умовах без автоматизації неможливо забезпечити ні своєчасний контроль, ні достатню точність, ні економічну ефективність заходів безпеки.

Автоматизація в хмарній безпеці охоплює кілька напрямів, кожен з яких має важливе значення для зниження ризиків і підвищення інтегрального рівня захищеності.

Першим ключовим напрямом є автоматизація перевірок відповідності конфігурацій (configuration compliance). Хмарні сервіси надають інтерфейси для автоматичного аналізу стану ресурсів: у AWS це Security Hub, Config та Inspector; в Azure - Defender for Cloud; у Google Cloud - Security Command Center. Такі інструменти дають змогу перевіряти сотні параметрів конфігурацій у реальному часі або з мінімальною затримкою та одразу повідомляти про відхилення від політик безпеки. Це принципово змінює підхід до контролю: організація не чекає періодичного аудиту, а має постійну картину стану захищеності.

Другим напрямом є автоматизація усунення порушень (auto-remediation). Йдеться не просто про виявлення помилки, а про її автоматичне виправлення. Наприклад, система може автоматично закривати публічний доступ до S3-бакета, обмежувати правила у Security Group або блокувати використання небезпечних IAM-політик. У практиці це значною мірою знижує показники MTTD і MTTR, а також усуває проблему “забутих” налаштувань, які залишилися після тестування чи тимчасових робочих завдань. Завдяки автоматизації організація рухається до моделі безпеки, де людський фактор має мінімальний вплив на критичні зони.

Третім напрямом є інтеграція безпеки у процеси розробки, або DevSecOps. Автоматичне сканування інфраструктури як коду (Terraform, CloudFormation), перевірка контейнерних образів, блокування деплою, що порушує політики, та підключення захисних механізмів у CI/CD-пайплайни - це необхідні складові того, що називають “безпекою за замовчуванням”. Таким чином, автоматизація дозволяє виявляти помилки ще на етапі розробки, коли їх усунення є дешевшим і менш ризикованим.

Не менш важливим аспектом є централізація логування та автоматичний аналіз подій. Завдяки сервісам на кшталт AWS CloudTrail, Azure Monitor чи Google Cloud Logging організація може збирати повний журнал подій, а за допомогою SIEM/SOAR-платформ - автоматично визначати аномалії, корелювати події та запускати сценарії реагування. Це особливо важливо з огляду на метрики з розділу 2: автоматизований аналіз журналів дозволяє суттєво зменшити MTTD (середній час виявлення) та покращити точність виявлення інцидентів.

У розрізі інтегральної моделі ефективності автоматизація безпосередньо впливає на низку ключових показників:

- Automation Level - рівень виконання контролів без участі людини;
- Compliance Rate - автоматичні перевірки підвищують дотримання внутрішніх стандартів;
- Coverage Index - автоматизовані інструменти дозволяють охопити більший обсяг ресурсів та сценаріїв;
- MTTD і MTTR - автоматизація аналізу та реагування скорочує обидва показники;

- Residual Risk - поєднання auto-remediation і CI/CD-сканування знижує залишковий ризик;
- ROSI - скорочується час фахівців, зменшується потреба у ручному моніторингу та аудитах.

Для сучасних хмарних інфраструктур автоматизація вже не є додатковою опцією - це вимога, без якої побудувати стійку до інцидентів і економічно ефективну систему безпеки практично неможливо. Саме тому провайдери активно рухаються у цьому напрямі: створюють API для всіх дій, впроваджують нативні сервіси auto-remediation, додають політики безпеки “за замовчуванням” та інтегрують механізми контролю в CI/CD-цикли.

3.1.4. Стандартизація та уніфікація конфігурацій (CIS, NIST, власні профілі)

Стандартизація конфігурацій є одним з найбільш дієвих і водночас недооцінених механізмів забезпечення безпеки у хмарному середовищі. На відміну від класичних локальних інфраструктур, де ресурси змінюються відносно повільно й часто налаштовуються вручну, у хмарі ключовим викликом є саме масштаб та швидкість змін. Тому єдиний спосіб зберегти контроль над конфігураціями - це впровадження системи стандартизованих профілів, які визначають дозволені параметри, способи розгортання, правила доступу й вимоги до логування.

Основним завданням стандартизації є не просто “уніфікувати налаштування”, а створити керовану, відтворювану та перевіряльну модель конфігурування, що дозволяє легко виявляти та виправляти відхилення. Тому у хмарних середовищах стандартизація є одночасно технічною та організаційною практикою.

Найпоширенішими базовими джерелами для побудови таких профілів є:

- CIS Benchmarks - докладні рекомендації для AWS, Azure, GCP та Kubernetes;
- NIST SP 800-53 та SP 800-207 (Zero Trust) - контрольні вимоги до архітектури і конфігурацій;
- Стандарти ISO/IEC 27017 та 27018 - орієнтовані на хмарну безпеку й захист даних;
- OWASP Cloud Security Guidelines - проникнення безпеки у DevOps-процеси;
- Власні профілі організації, що враховують бізнес-потреби, ризики та архітектурні особливості.

Стандартизовані профілі конфігурацій виконують декілька важливих функцій:

1. Формують єдину базову лінію (baseline) для всіх ресурсів.

Це означає, що кожний новий ресурс - S3-бакет, EC2-екземпляр, база даних, контейнерний кластер - створюється лише відповідно до затверджених параметрів. Такий підхід мінімізує кількість “ручних” налаштувань і, відповідно, людських помилок.

2. Дозволяють автоматизувати аудит і перевірки.
Оскільки політики чітко описані, інструменти на кшталт AWS Config, Azure Policy, Terraform Compliance або Open Policy Agent (OPA) можуть автоматично перевіряти, чи відповідає конфігурація встановленим вимогам. Це напряму підвищує Compliance Rate та зменшує залишковий ризик.
3. Підтримують масштабованість і уніфікацію.
У великих середовищах (100+ акаунтів або десятки команд розробки) стандартизація дозволяє уникнути "зоопарку" різних підходів і налаштувань, які утворюються без чітких правил.
4. Сприяють впровадженню концепцій DevSecOps.
Коли конфігурація описана у вигляді коду (IaC), стандартизовані профілі легко включаються в CI/CD-профілі, блокуючи некоректні зміни на ранніх етапах.
5. Забезпечують прозорість та легкість проведення аудиту.
Аудитор може перевіряти відповідність не вручну, а автоматично, порівнюючи конфігурації з базовими профілями.

У практичному застосуванні стандартизація значно підвищує такі метрики з розділу 2:
- Compliance Rate, оскільки всі ресурси оцінюються щодо чітко визначених правил;
- Coverage Index, оскільки профілі охоплюють широку номенклатуру сервісів;
- Automation Level, завдяки можливості автоматизованих перевірок і auto-remediation;
- Detection Accuracy, оскільки чіткі правила дозволяють точніше визначати відхилення;
- Residual Risk, який знижується за рахунок стабільності та передбачуваності конфігурацій.

Важливо й те, що стандартизація не обмежується використанням існуючих профілів (наприклад, CIS). У більшості випадків організації створюють гібридні профілі, що базуються на:
- готових стандартах (CIS, NIST),
- специфічних вимогах бізнесу (наприклад, обов'язкове шифрування ключами KMS),
- вимогах комплаєнсу (PCI DSS, GDPR),
- архітектурних особливостях (серверniless або контейнерні сервіси).

3.1.5. Побудова безпечної архітектури (Zero Trust, segmentation, network isolation)

Побудова безпечної архітектури є фундаментальним елементом хмарної безпеки, оскільки саме архітектурні рішення визначають межі можливих ризиків і формують основу, на якій працюють усі інші засоби захисту. У хмарному середовищі архітектура не є статичною - вона постійно змінюється разом із масштабуванням, впровадженням нових сервісів та автоматизацією процесів.

Тому безпека повинна бути інтегрована в архітектуру за замовчуванням, а не додана після розгортання.

У цьому контексті особливе значення мають три концепції: Zero Trust, мережева сегментація та ізоляція середовищ. Кожна з них доповнює одна одну і разом формує багаторівневу модель захисту, здатну протистояти як зовнішнім атакам, так і внутрішнім загрозам.

Zero Trust як архітектурна парадигма

Zero Trust - це не окремий продукт чи технологія, а концепція, що виходить із принципу: “нікому не довіряй за замовчуванням, перевіряй завжди”. На відміну від традиційної моделі, де мережі зсередини вважалися відносно безпечними, Zero Trust передбачає постійну перевірку кожного запиту - незалежно від його походження.

У хмарному контексті принцип Zero Trust реалізується через:

- обов’язкову автентифікацію та авторизацію кожного звернення (machine-to-machine included);
- перевірку контексту запиту (“context-aware authorization”);
- мікросегментацію мережевої взаємодії;
- мінімізацію довірених зон (no “trusted internal network”);
- постійну перевірку конфігурацій та аномалій доступу.

Завдяки цьому Zero Trust значно підвищує такі метрики, як Detection Accuracy, Reduction of Residual Risk та Compliance Rate, оскільки створює передумови для контролю кожного рівня взаємодії.

Мережева сегментація та контроль комунікацій

Другим компонентом є мережева сегментація - поділ інфраструктури на ізольовані підмережі з мінімально необхідними взаємодіями між ними. У хмарних сервісах це зазвичай реалізується через:

- VPC та підмережі (public/private),
- route tables та network ACLs,
- security groups,
- окремі VPC для різних середовищ,
- приватні канали взаємодії (VPC Endpoints, Private Link).

Сегментація дозволяє зменшити “площу атаки”: навіть якщо один компонент буде скомпрометований, зловмисник не зможе вільно переміщуватися всередині мережі. Це також створює архітектурну основу для Zero Trust, оскільки контроль доступу відбувається на кількох рівнях.

У практичній площині сегментація впливає на Coverage Index (охоплення контролів), Compliance Rate (CIS і NIST прямо вимагають сегментації) та Residual Risk (різко зменшується можливість lateral movement).

Ізоляція робочих середовищ

Окрім мережевої сегментації, важливим напрямом є логічна та операційна ізоляція середовищ - development, testing, staging, production. В ідеалі кожне середовище повинно мати окремі:

- акаунти (наприклад, окремий AWS account на кожне середовище);
- мережеві сегменти;
- ключі шифрування;
- IAM-політики;
- контрольні групи та політики безпеки.

Відокремлення середовищ мінімізує ризики випадкового або несанкціонованого внесення змін - особливо у виробничому середовищі. Воно також безпосередньо впливає на економічну ефективність: інциденти в ізольованих середовищах мають набагато менший вплив на бізнес, отже зменшують витрати на ліквідацію наслідків.

Архітектурні рекомендації у хмарних платформах

Практичні приклади архітектурних патернів у хмарних платформах включають:

- AWS: використання AWS Organizations, Service Control Policies, VPC Endpoints, PrivateLink, Security Hub як централізованого контролера.
- Azure: застосування Management Groups, Azure Policy, Virtual Network Service Endpoints, Defender for Cloud.
- GCP: поділ проектів, використання Shared VPC, IAM service perimeter (VPC-SC), Cloud Armor.

Усі ці платформи рухаються у бік Zero Trust і мікросегментації як базового стандарту, оскільки хмарні середовища все частіше стають ціллю атак, спрямованих не на периметр, а на внутрішнє переміщення.

Зв’язок із метриками ефективності

Впровадження безпечної архітектури має суттєвий вплив на інтегральні показники безпеки (SPS):

- підвищує Compliance Rate, оскільки більшість архітектурних вимог стандартизовані;
- зменшує залишковий ризик, оскільки розрив зв’язків між компонентами блокує розвиток атаки;
- збільшує Coverage Index, оскільки більше ресурсів охоплено чіткими політиками;
- покращує Detection Accuracy, бо логування та аналіз подій здійснюються на кількох рівнях;
- підтримує низькі MTTD і MTTR, оскільки аномалії легше ідентифікувати.

Таким чином, безпечна архітектура є стратегічною основою для побудови стійкого хмарного середовища, а її принципи будуть безпосередньо враховані у рекомендаціях наступних підрозділів та в описі прикладної утиліти для аналізу AWS.

3.2. Технічні рекомендації щодо підвищення рівня безпеки у хмарних середовищах

3.2.1. Удосконалення ідентифікації та автентифікації

Надійна система ідентифікації та автентифікації є основою будь-якої моделі безпеки у хмарних середовищах. Саме на цьому рівні визначається, які суб'єкти можуть виконувати операції з ресурсами, які привілеї їм надаються та як контролюється їхня взаємодія. З огляду на відсутність традиційного мережевого периметра, хмарні середовища значною мірою грунтуються на механізмах доступу, пов'язаних із обліковими записами, ключами та ролями. Тому підвищення якості ідентифікації та автентифікації є одним із ключових напрямів забезпечення ефективного захисту.

Одним із основних інструментів удосконалення автентифікації є впровадження багатофакторної автентифікації (MFA) для всіх видів доступу - як користувацького, так і адміністративного. MFA суттєво зменшує ризик несанкціонованого доступу, особливо у випадках, коли облікові дані були скомпрометовані. Практика провайдерів свідчить, що атаки на хмарні акаунти часто спираються саме на вкрадені або ненадійні креденшали; тому запровадження MFA є одним із найбільш ефективних контролів, який безпосередньо впливає на зниження ризиків.

Другим важливим напрямом є поступовий перехід до passwordless-автентифікації. Використання апаратних ключів (наприклад, FIDO2), а також механізмів автентифікації на основі сертифікатів дозволяє усунути низку проблем, пов'язаних із паролями: їхнє повторне використання, недостатню складність або фішингові атаки. Хмарні платформи вже підтримують такі моделі, надаючи можливість інтеграції як з корпоративними каталогами, так і з зовнішніми провайдерами ідентичностей.

Окрему увагу слід приділити керуванню ключами доступу та секретами. У хмарних середовищах значна частина взаємодій відбувається через API, а тому програмні токени мають такі самі ризики, як і традиційні логіни користувачів. Ключовими рекомендаціями є:

- відмова від статичних або довготривалих ключів;
- використання короткоживучих токенів (STS, OIDC);
- зберігання секретів у спеціалізованих сховищах (AWS Secrets Manager, Azure Key Vault, HashiCorp Vault);
- регулярна ротація ключів та автоматичний контроль їх використання;
- обмеження ключів прив'язаними до ролей і чіткими політиками.

Важливо також приділити увагу централізованому управлінню ідентичностями (Identity Governance). У великих організаціях наявність різнорідних доступів і ролей створює значний ризик їх дублювання або накопичення надмірних прав. Використання каталогу ідентичностей (Azure AD, AWS IAM Identity Center тощо) забезпечує уніфікацію та дозволяє

впроваджувати життєвий цикл доступів - створення, зміна, тимчасове привілейоване підвищення, скасування.

Удосконалення механізмів ідентифікації та автентифікації також безпосередньо впливає на ключові метрики, визначені у розділі 2. Зокрема:

- підвищується рівень відповідності (Compliance Rate), адже вимоги до автентифікації прямо регламентовані більшістю стандартів безпеки;
- зменшується залишковий ризик, оскільки компрометація одного компонента має набагато менше шансів призвести до масштабного порушення;
- покращується точність виявлення (Detection Accuracy) завдяки централізованому аудиту автентифікаційних подій;
- скорочуються MTTD і MTTR, адже аномалії в доступі фіксуються одразу на рівні інфраструктури;
- підвищується Automation Level, коли ротація ключів, контроль привілеїв та політики автентифікації впроваджуються автоматизовано.

Таким чином, удосконалення системи ідентифікації та автентифікації є базовим етапом зміцнення хмарної безпеки. Воно створює основу для застосування інших технічних контролів і дозволяє суттєво підвищити ефективність системи захисту в цілому.

3.2.2. Посилення контролю доступу (IAM, SCP, permission boundaries)

Контроль доступу є одним із ключових механізмів забезпечення безпеки у хмарних середовищах, оскільки саме він визначає межі дозволеної взаємодії суб’єктів з ресурсами. Неправильне або надмірно широке надання привілеїв є однією з найпоширеніших причин хмарних інцидентів, що підтверджують як статистичні звіти, так і результати інцидент-розслідувань. Тому удосконалення механізмів контролю доступу є важливим технічним завданням, яке напряму впливає на загальний рівень безпеки.

У хмарних платформах система контролю доступу, як правило, реалізується за допомогою моделі ролей і політик (IAM). Її ефективність залежить не лише від правильності написання політик, але й від того, наскільки чітко визначено межі дозволених операцій.

Першим кроком у посиленні контролю доступу є оптимізація IAM-політик. Це включає:

- уникнення використання широких операторів типу "*" у дозволах;
- чітке визначення дій (Actions) і ресурсів (Resources) для кожної політики;
- розділення політик на дрібніші елементи, що відповідають конкретним функціям;
- впровадження окремих ролей для сервісів, користувачів, CI/CD-процесів та автоматизованих завдань.

У реальних хмарних середовищах часто зустрічається накопичення “тимчасових” або “запасних” дозволів, доданих для вирішення оперативних задач. Тому рекомендовано проводити регулярний аналіз фактичного

використання дозволів, що дозволяє виявляти невикористані привілеї й автоматично оптимізувати політики.

Важливою складовою посилення контролю доступу є використання Service Control Policies (SCP) у середовищах, де задіяні кілька акаунтів або діє централізована модель управління (наприклад, AWS Organizations, Azure Management Groups). SCP дозволяють визначити глобальні обмеження, що застосовуються до всіх акаунтів або груп акаунтів, незалежно від того, які IAM-політики призначені всередині конкретного середовища. Це створює додатковий рівень захисту, який унеможливлює виконання критично небезпечних операцій, таких як вимкнення логування, видалення ключів KMS або створення публічних ресурсів.

Ще одним важливим інструментом є permission boundaries - механізм, який визначає верхню межу привілеїв для ролей чи користувачів. Permission boundaries дозволяють делегувати створення політик без ризику надання надмірних прав, оскільки будь-які дозволи автоматично обмежуються заданими межами. Цей механізм є корисним у великих організаціях, де частину прав потрібно делегувати командам або сервісним акаунтам, але необхідно уникнути неконтрольованого розширення привілеїв.

У рекомендаціях щодо посилення контролю доступу також важливо зазначити:
- прив’язку ролей до робочого навантаження (workload identity), а не до статичних ключів;
- розмежування доступу між середовищами (dev/test/prod);
- використання окремих ролей для різних сервісів замість повторного застосування універсальних;
- автоматизацію створення, оновлення та перевірки ролей через IaC.

З точки зору метрик ефективності, посилення контролю доступу впливає на:
- Compliance Rate, оскільки більшість стандартів безпеки висувають строгі вимоги до управління привілеями;
- Detection Accuracy, через точнішу реєстрацію операцій за кожною роллю;
- Residual Risk, який суттєво знижується при мінімізації надмірних дозволів;
- Automation Level, коли створення та перевірки IAM автоматизуються;
- Coverage Index, оскільки правильна архітектура доступів забезпечує контроль над усіма типами ресурсів.

У сукупності ці рекомендації створюють системний підхід до управління доступом у хмарному середовищі. Він дозволяє обмежити потенційні вектори атак, зменшити вплив людського фактора та підвищити стійкість інфраструктури без збільшення операційного навантаження на команду.

3.2.3. Безпечна конфігурація мережі

Мережева безпека у хмарних середовищах суттєво відрізняється від традиційних підходів, характерних для локальних дата-центрів. В умовах хмари мережа є логічною конструкцією, яка масштабуються програмно та має

розподілену природу. Це змінює як можливості, так і ризики, роблячи правильну конфігурацію мережі критичним компонентом загальної безпеки. Основними напрямами формування безпечної мережевої архітектури є сегментація, використання приватних каналів зв'язку, контроль трафіку та захист веб-рівня.

Сегментація мережі (VPC segmentation)
Сегментація є фундаментом хмарної мережевої безпеки. Створення окремих віртуальних приватних хмар (VPC), поділ їх на приватні та публічні підмережі, а також управління маршрутизацією створюють ізольовані домени, що обмежують взаємодію сервісів.

У практиці це передбачає:
- відокремлення публічних ресурсів (наприклад, балансувальників) від приватних (бази даних, обчислювальні сервери);
- розділення різних типів робочих навантажень на окремі підмережі або навіть окремі VPC;
- застосування окремих ACL та маршрутних таблиць для підвищення granularності контролю.

Завдяки сегментації суттєво зменшується площа атаки, а ризики lateral movement (бокового переміщення атакувальника) мінімізуються. Це безпосередньо відображається на зниженні залишкового ризику та підвищенні загальної стійкості системи.

Використання приватних endpoint'ів (Private Endpoints, VPC Endpoints)
Однією з основних рекомендацій сучасних провайдерів є уникнення передачі трафіку через публічні мережі, навіть якщо доступ відбувається до "хмарних" сервісів. Використання приватних endpoint'ів дозволяє направляти весь внутрішній трафік через захищену приватну мережу провайдера.

Приклади таких механізмів:
- AWS PrivateLink / VPC Endpoints
- Azure Private Endpoints
- Google Private Service Connect

Це не лише зменшує ризик перехоплення трафіку, але й дозволяє створювати повністю приватні архітектури, де застосунки не мають виходу в публічний інтернет, що відповідає принципам Zero Trust.

Контроль трафіку (Security Groups, Network ACLs, Firewalling)
У хмарі контроль трафіку здійснюється на кількох рівнях:
1. Security Groups - керують трафіком на рівні інстансу або сервісу;
2. Network ACLs - контролюють трафік на рівні підмережі;
3. Firewall-політики провайдерів або сторонніх рішень - забезпечують глибоку перевірку та аналіз.

Комплексне застосування цих механізмів дозволяє:
- обмежити доступ лише до необхідних портів;
- суворо контролювати inbound/outbound-трафік;
- уникати необмежених правил, таких як 0.0.0.0/0;
- застосовувати whitelisting замість загальних дозволів.

Це напряму впливає на Coverage Index, оскільки покриває більшість мережевих взаємодій, та на Compliance Rate, оскільки вимоги до firewall-контролю прямо визначені у CIS і NIST.

Захист веб-рівня (WAF, CDN, бот-захист)

Оскільки веб-додатки є одним з найпоширеніших векторів атак, важливим елементом мережевої безпеки є веб-application firewall (WAF). У хмарі це зазвичай реалізується як керована служба:
- AWS WAF
- Azure Web Application Firewall
- Google Cloud Armor

WAF забезпечує захист від:
- SQL-injection;
- XSS;
- RCE та інших застосункових атак;
- бот-трафіку та DDoS.

У поєднанні з CDN це дозволяє не лише зменшити ризики атак, але й покращити продуктивність. Наявність WAF може також стати складовою автоматизованого підвищення точності виявлення аномалій у середовищі.

Зв’язок з метриками ефективності

Рекомендації щодо мережевої безпеки істотно впливають на:
- Residual Risk - ізоляція та фільтри знижують вплив можливих порушень;
- Coverage Index - мережеві правила охоплюють усі потоки даних;
- Detection Accuracy - комбінований моніторинг дозволяє точніше виявляти атаки;
- Compliance Rate - CIS та NIST вимагають сегментації й контрольованого трафіку;
- MTTD і MTTR, коли WAF і firewall-логування інтегровані з SIEM/SOAR.

У сукупності безпечна конфігурація мережі формує “транспортний” рівень захисту, який працює разом із IAM, архітектурними принципами та автоматизацією. Якісно налаштована мережа не лише зменшує ризики, а й робить середовище більш передбачуваним і легко контрольованим, що є важливою передумовою для подальшого розгортання комплексних рішень безпеки.

3.2.4. Шифрування даних у стані спокою і під час передавання

Шифрування є одним із ключових механізмів захисту інформації у хмарних середовищах, оскільки воно забезпечує захист даних навіть у випадках, коли інші засоби безпеки були обійдені. Хмарні платформи пропонують широкий спектр вбудованих інструментів для шифрування даних у стані спокою (at rest) та під час передавання (in transit), а ефективне використання цих інструментів є критично важливим як для технічної безпеки, так і для відповідності нормативним вимогам.

Шифрування даних у стані спокою (at rest)

У стані спокою дані можуть зберігатися у сховищах об’єктів, файлових системах, базах даних, кешах або резервних копіях. Хмарні провайдери надають механізми прозорого шифрування, що можуть бути ввімкнені без змін у застосунку. Сюди входить:

- AWS: KMS, EBS encryption, S3 Server-Side Encryption (SSE), RDS encryption;
- Azure: Azure Key Vault, Storage Service Encryption, Transparent Data Encryption;
- GCP: Cloud KMS, CMEK, CSEK, шифрування Cloud Storage.

Хоча більшість сервісів підтримують шифрування “за замовчуванням”, важливо правильно обирати типи ключів:

- KMS-managed keys (AWS-managed, platform-managed) - зручні, але менш контрольовані;
- customer-managed keys (CMK/CMEK) - забезпечують повний контроль над життєвим циклом ключів;
- customer-supplied keys - найвищий рівень контролю, але складність управління.

Застосування власних ключів (CMEK/CMK) дозволяє реалізувати політики ротації, обмеження географії ключів, розмежування прав та незалежний аудит доступу.

Рекомендовано також:

- включити шифрування для абсолютно всіх типів сховищ (включно з тимчасовими);
- обмежувати доступ до ключів через IAM-політики та ключові політики (key policies);
- проводити регулярну ротацію ключів та моніторити їх використання;
- використовувати окремі ключі для різних середовищ або груп даних.

Шифрування під час передавання (in transit)

Передавання даних через мережеві канали є одним з найбільш вразливих етапів взаємодії у хмарному середовищі. Хоча багато хмарних сервісів автоматично підтримують TLS, забезпечення безпеки трафіку вимагає додаткових заходів.

Основні рекомендації включають:
- використання TLS 1.2 або 1.3 у всіх внутрішніх та зовнішніх з’єднаннях;
- застосування зашифрованих протоколів для API-взаємодії;
- конфігурацію ALB/ELB/CloudFront для примусового використання HTTPS;
- впровадження mTLS (mutual TLS) у сервісних або мікросервісних архітектурах;
- блокування незашифрованих протоколів та портів на рівні Security Groups/Firewall.

У розподілених застосунках з великою кількістю сервісів рекомендується впроваджувати service mesh (наприклад, AWS App Mesh, Istio), який забезпечує шифрування трафіку між сервісами без змін у коді.

Управління ключами (Key Management) як складова безпеки
Шифрування неможливе без ефективного управління ключами. Недостатній контроль над ключами може повністю звести нанівець переваги шифрування. Основні принципи:
- зберігання ключів виключно у керованих сервісах (KMS, Secrets manager, Key Vault, Cloud KMS);
- обмеження доступу до операцій “Decrypt” і “GenerateDataKey”;
- розділення ролей адміністрування ключів (key admins) та користувачів ключів (key users);
- аудит та журналювання всіх операцій з ключами;
- увімкнення автоматичної ротації ключів (за можливості років - 1 рік або менше).

Належна організація key management практично гарантує, що навіть у разі доступу до зашифрованих даних зловмисник не зможе їх розшифрувати.

Вплив на метрики ефективності
Шифрування та управління ключами суттєво впливають на:
- Compliance Rate, оскільки більшість стандартів (GDPR, PCI DSS, ISO 27018) прямо вимагають шифрування;
- Residual Risk, який суттєво знижується, навіть якщо зловмисник отримав доступ до середовища;
- Coverage Index, оскільки шифрування охоплює всі типи даних;
- Detection Accuracy, завдяки журналюванням операцій з ключами;
- Security Posture Score, що підвищується через правильну організацію даних і ключів.

У підсумку шифрування даних у стані спокою та під час передавання є критично важливим елементом безпеки хмарних платформ. Його ефективність значною мірою залежить від того, наскільки якісно реалізовані процеси управління ключами та наскільки повно охоплені всі типи даних і каналів зв’язку.

3.2.5. Контроль конфігурацій та IaC-сканування (Terraform scans, OPA/Rego, CI/CD)

Управління конфігураціями у хмарному середовищі є складним завданням через високу динамічність ресурсів, часті зміни та багатокомпонентність сучасних систем. Відсутність централізованого контролю або узгоджених процесів створює передумови для помилок конфігурацій, які, згідно з дослідженнями провідних аналітичних компаній, є одним з найпоширеніших джерел хмарних інцидентів. Тому контроль конфігурацій та інтеграція безпеки в інфраструктуру як код (Infrastructure as Code, IaC) відіграють важливу роль у забезпеченні стабільного та керованого середовища.

Автоматизований контроль конфігурацій

У хмарних платформах автоматизований контроль конфігурацій забезпечують такі інструменти, як:

- AWS Config - контроль стану ресурсів, виявлення відхилень, автоматичні дії;
- Azure Policy - застосування політик до підписок та груп управління;
- Google Config Validator / Organization Policy;
- сторонні платформи на кшталт Prisma Cloud, Wiz, Lacework, що аналізують конфігурації на міжхмарному рівні.

Принцип роботи таких систем полягає у постійній оцінці фактичної конфігурації порівняно з політиками або стандартами (CIS, NIST, власні профілі). У разі порушення система генерує подію, яка може бути передана до SIEM або використана для автоматичного виправлення.

Перевага цього підходу полягає у тому, що контроль конфігурації не залежить від окремого інженера чи команди - він працює постійно та є уніфікованим для всього середовища.

IaC як основа стабільності та передбачуваності

Використання інфраструктури як коду (IaC) значно зменшує ризики, пов’язані з ручними налаштуваннями. Terraform, CloudFormation, Pulumi та інші інструменти дозволяють:

- документувати конфігурації у вигляді коду;
- впроваджувати контроль версій (Git);
- застосовувати стандартизацію через модулі та шаблони;
- автоматизувати розгортання.

IaC забезпечує повну відтворюваність, що дозволяє уникнути “дрейфу” конфігурацій - розсинхронізації між тим, що визначено в коді, і тим, що існує насправді.

IaC-сканування: Terraform scans

Окремим напрямом контролю є IaC-сканування - аналіз конфігурацій до їх розгортання. Інструменти сканування дозволяють виявити помилки на ранньому

етапі, зменшуючи імовірність потрапляння некоректної конфігурації у production. Найпоширенішими інструментами є:

- Checkov (Bridgecrew/Palo Alto)
- tfsec / Trivy
- Terrascan
- Sentinel у Terraform Enterprise
- kics, Regula та інші.

Ці інструменти аналізують Terraform-файли, порівнюючи їх із набором правил (наприклад, “заборонено відкриті S3-бакети”, “EC2 повинен бути у приватній підмережі”, “KMS-ключі мають бути увімкненими”).

IaC-сканування істотно знижує залишковий ризик, оскільки унеможливлює появу вразливих налаштувань у реальному середовищі.

Open Policy Agent (OPA) і Rego як механізм формалізації політик

Open Policy Agent (OPA) - універсальний рушій політик, який дозволяє формалізувати правила безпеки у вигляді декларативного коду на мові Rego. Його застосування поширюється на:

- Kubernetes admission control;
- валідацію Terraform;
- політики в CI/CD;
- контроль доступу у сервісах.

OPA робить політики прозорими, перевіряльними та придатними для автоматизації. Наприклад, політика може вимагати, щоб кожен Terraform-модуль використовував CMK-ключ для шифрування або щоб усі Security Groups мали whitelist-підхід.

Інтеграція з CI/CD-процесами

Контроль конфігурацій досягає найбільшої ефективності тоді, коли інтегрується у CI/CD-пайплайни. Автоматизація на цьому рівні дозволяє:

- блокувати деплой, якщо конфігурація порушує політику;
- проводити IaC-сканування під час кожного Pull Request;
- додавати підписи (approvals) для критичних змін;
- автоматично генерувати звіти про стан інфраструктури.

Запуск сканерів як частини GitHub Actions, GitLab CI або Jenkins дозволяє гарантувати, що жодна зміна не потрапить у хмару без перевірки.

Вплив на метрики ефективності

Контроль конфігурацій та IaC-сканування впливають практично на всі ключові метрики, розроблені у розділі 2:

- Compliance Rate - автоматизовані перевірки забезпечують постійну відповідність;
- Coverage Index - автоматизація охоплює всі ресурси, включно з тими, що лише плануються;

- Automation Level - сканери і OPA забезпечують повну автоматизацію контролю;
- MTTD/MTTR - виявлення помилок відбувається миттєво, ще до застосування змін;
- Residual Risk - модель IaC різко зменшує ризик “дрейфу конфігурацій”;
- SPS - інтегральний показник підвищується завдяки стабільності та контрольованості.

Таким чином, контроль конфігурацій та IaC-сканування формують основу передбачуваної та стійкої хмарної архітектури. Це один із найдієвіших способів зменшення ризиків, оскільки він забезпечує автоматичний захист не лише на експлуатаційному рівні, але й на етапі проєктування та розробки.

3.2.6. Удосконалення логування та моніторингу

Логування та моніторинг є критичними елементами системи хмарної безпеки, які забезпечують можливість своєчасного виявлення інцидентів, аналізу подій та відтворення хронології дій у середовищі. На відміну від традиційних інфраструктур, хмарні середовища характеризуються високою динамічністю, автоматизацією та великою кількістю взаємодій між сервісами. У таких умовах ефективний моніторинг неможливий без системного підходу до збору журналів, їх агрегації, аналізу та автоматизованого реагування.

Централізоване логування у хмарних середовищах
Першим завданням є забезпечення повного та централізованого збору логів. Хмарні провайдери пропонують нативні сервіси, серед яких:

- AWS CloudTrail - реєстрація всіх API-викликів та дій користувачів;
- AWS CloudWatch Logs / Logs Insights - збір системних логів та телеметрії;
- Azure Monitor / Activity Logs;
- Google Cloud Audit Logs.

Централізоване логування дає змогу:

- забезпечити безперервний аудит операцій;
- виявляти аномальні або нехарактерні дії;
- фіксувати факти успішної чи невдалої автентифікації;
- проводити ретроспективний аналіз інцидентів.

У більшості випадків важливо не лише збирати логи, але й зберігати їх у немодифікованому вигляді (immutable storage), наприклад у версіонованих S3-бакетах або за допомогою Vault Lock (WORM-режим).

Аналітика логів і виявлення аномалій
Збирання логів саме по собі не дає безпеки - необхідний аналіз. Хмарні сервіси increasingly інтегрують механізми виявлення аномалій:

- AWS GuardDuty - машинне навчання для виявлення підозрілих дій;
- Azure Defender for Cloud - вбудовані аналітичні моделі;
- GCP Event Threat Detection - виявлення аномалій та шаблонів атак.

Такі системи аналізують:
- API-виклики, які виходять за межі звичайної поведінки;
- нетипові географічні локації;
- доступ до конфіденційних сервісів;
- підозрілий мережевий трафік;
- взаємодії з вразливими або публічними ресурсами.

Використання ML/AI-моделей значно покращує якість виявлення інцидентів (Detection Accuracy), особливо у великих середовищах, де ручний аналіз є неможливим.

Інтеграція логів у SIEM-системи
Для комплексного моніторингу великі організації використовують SIEM-платформи, такі як:
- Splunk,
- IBM QRadar,
- Elastic Security,
- Azure Sentinel,
- Securonix,
- Orca,
- ArcSight.

Інтеграція хмарних логів у SIEM дає можливість:
- корелювати події з різних джерел;
- створювати кастомні правила виявлення;
- будувати сценарії реагування;
- виконувати глибокий аналіз інцидентів.

У SIEM також формується єдиний “центр спостереження” для команди SOC.

SOAR і автоматизоване реагування
Наступним рівнем є використання SOAR-платформ, які дозволяють автоматизувати реагування на інциденти. У контексті хмари SOAR може:
- блокувати облікові записи;
- закривати мережеві порти;
- вимикати IAM-ключі;
- створювати квитки інцидентів;
- запускати скрипти для auto-remediation.

Завдяки SOAR значно зменшуються показники:

- MTTD (mean time to detection);
- MTTR (mean time to response).

Обов’язкові налаштування для підвищення безпеки
Щоб логування і моніторинг були ефективними, рекомендується:
- увімкнути CloudTrail у всіх регіонах та всіх акаунтах;
- використовувати multi-account організації для централізованого збору;
- створити окремий “audit” акаунт із обмеженим доступом;
- увімкнути VPC Flow Logs та DNS Logs;
- аналізувати доступ до S3, KMS, IAM та інших критичних сервісів;
- впровадити логування на рівні застосунків та контейнерів.

У поєднанні з SIEM/SOAR цей підхід створює багаторівневу систему моніторингу, яка охоплює як інфраструктуру, так і додатки.

Вплив на метрики ефективності
Удосконалення логування та моніторингу впливає майже на всі ключові показники з розділу 2:
- MTTD - різко скорочується завдяки автоматизованому аналізу;
- MTTR - знижується за рахунок SOAR і узгоджених playbooks;
- Detection Accuracy - підвищується завдяки ML/AI-аналітиці;
- Coverage Index - охоплює всі рівні архітектури, від мережі до застосунку;
- Residual Risk - зменшується, оскільки інциденти виявляються до їхнього розвитку;
- Compliance Rate - більшість стандартів прямо вимагають логування та аналіз.

У підсумку логування та моніторинг є не лише інструментами спостереження, але й однією з основних складових активної хмарної безпеки. Правильно побудована система логування дає змогу своєчасно виявляти інциденти, оцінювати стан середовища та формувати реагування, що є ключовим для підвищення загальної ефективності захисту.

Висновки

У третьому розділі було сформовано комплекс практичних рекомендацій, спрямованих на підвищення ефективності захисту інформації у хмарних середовищах. Вони грунтуються на виявлених у розділі 1 недоліках сучасних підходів та на формалізованій системі критеріїв і метрик, розроблених у розділі 2. Сукупність запропонованих заходів охоплює технічний, організаційний та процесний рівні безпеки і забезпечує системний підхід до вдосконалення хмарної інфраструктури.

У межах розділу було визначено ключові принципи, на яких базується ефективна модель хмарної безпеки. До них належать: принцип найменших привілеїв, багаторівневий захист, автоматизація безпекових процесів, стандартизація конфігурацій та побудова архітектури відповідно до концепцій Zero Trust та ізоляції середовищ. Ці принципи створюють фундамент, що дозволяє впроваджувати практичні заходи у спосіб, який забезпечує стійкість, передбачуваність та узгодженість роботи хмарної інфраструктури.

Подальша частина розділу була присвячена технічним рекомендаціям, які деталізують конкретні способи реалізації принципів безпеки. Зокрема, детально розглянуто вдосконалення механізмів ідентифікації та автентифікації, посилення контролю доступу за допомогою IAM, SCP та permission boundaries, побудову безпечної мережевої конфігурації на основі сегментації й приватних каналів, застосування шифрування даних у стані спокою та під час передавання, а також використання IaC-сканування та автоматизованого контролю конфігурацій для запобігання помилкам на ранніх етапах. Окрему увагу приділено вдосконаленню логування та моніторингу за рахунок централізованого збору журналів, аналітики на основі машинного навчання та впровадження SIEM/SOAR-рішень.

Важливо, що запропоновані заходи не існують ізольовано: вони взаємодоповнюють один одного та впливають на широкий спектр метрик ефективності, визначених у попередньому розділі. Такі показники, як Compliance Rate, Coverage Index, MTTD, MTTR та інтегральний показник SPS, напряму покращуються завдяки впровадженню рекомендованих практик. Це доводить, що підвищення рівня безпеки у хмарному середовищі є досяжним лише за умови системного підходу, де технічні засоби, процесні регламенти та автоматизація формують цілісну модель захисту.

Проте навіть найкращі рекомендації втрачають ефективність без надійних механізмів їх автоматизованої перевірки та контролю. Динаміка хмарних середовищ, часті оновлення ресурсів, значна кількість ролей та сервісів роблять ручний аудит малоефективним і непридатним для масштабування. Саме тому логічним продовженням розробленої системи рекомендацій є створення власного програмного засобу, здатного автоматично аналізувати стан AWS-акаунта, зіставляти його з розробленими критеріями та генерувати інтегральну оцінку безпеки на основі формалізованих метрик.

Таким чином, результати третього розділу не лише окреслюють шляхи підвищення ефективності захисту, але й формують практичні вимоги до інструмента, який дозволить застосувати рекомендації у реальному середовищі та забезпечити їх безперервний контроль. Це підводить до логічного переходу до четвертого розділу роботи, де буде представлено архітектуру, реалізацію та практичне застосування розробленої утиліти для автоматизованого аналізу хмарної безпеки.

Розділ 4. Розробка та практична реалізація утиліти для автоматизованого аналізу безпеки

У попередніх розділах було сформовано теоретичні засади та практичні рекомендації щодо підвищення рівня безпеки хмарних середовищ. На основі аналізу сучасних загроз, визначення критеріїв оцінювання та розробки системи метрик було продемонстровано, що ефективність захисту значною мірою залежить від здатності організації забезпечувати постійний контроль стану інфраструктури та своєчасно виявляти відхилення від рекомендованих політик. Саме динамічність хмарних платформ, велика кількість компонентів та природна мінливість конфігурацій створюють умови, за яких ручний аналіз стає непридатним як з погляду масштабованості, так і з погляду точності.

У цьому контексті особливого значення набувають автоматизовані рішення, здатні не лише збирати та аналізувати інформацію щодо поточного стану середовища, але й співвідносити ці дані з формалізованими критеріями та показниками, визначеними в другому розділі. Використання таких засобів дозволяє систематизувати процес оцінювання, зменшити вплив людського чинника, забезпечити передбачуваність результатів і водночас підвищити частку автоматизації у сфері хмарної безпеки.

Цей розділ присвячений розробці та опису утиліти для автоматизованого аналізу стану безпеки AWS-акаунта. На відміну від загальних теоретичних моделей, представлена утиліта є практичним втіленням принципів і рекомендацій, сформульованих у розділі 3. Вона поєднує механізми збору конфігураційних даних, набір структурованих правил перевірок та модуль обчислення метрик, включно з інтегральним показником SPS, що дозволяє кількісно оцінити ефективність реалізованих засобів захисту.

Побудова такої утиліти дає змогу продемонструвати не лише можливість автоматизації перевірок, але й те, як формалізована система показників може застосовуватися на практиці. Утиліта є прикладом інструмента, який можна інтегрувати в існуючі процеси аудитів, DevSecOps-пайплайни або системи моніторингу, забезпечуючи безперервний контроль і наочну оцінку стану безпеки.

У межах цього розділу буде розглянуто архітектуру запропонованого рішення, принципи побудови його основних модулів, набір правил, що реалізують ключові рекомендації з розділу 3, а також формат і логіку формування підсумкових звітів. Окрему увагу приділено тому, як результати роботи утиліти співвідносяться з метриками ефективності та демонструють практичну цінність розробленої методології.

4.1. Загальна схема роботи інструмента

Розроблена утиліта грунтується на модульному та поетапному підході до аналізу стану безпеки AWS-акаунта. Метою є забезпечення послідовного, відтворюваного й автоматизованого процесу оцінювання, який охоплює збір даних, їх інтерпретацію, застосування набору правил та формування інтегральних метрик. Загальна схема роботи інструмента може бути представлена як ланцюг взаємопов'язаних етапів, де кожен модуль виконує чітко визначену функцію, а результат одного етапу є вхідними даними для наступного.

1. Ініціалізація та зчитування конфігурації

Робота починається зі зчитування YAML-конфігураційного файлу, у якому визначено параметри підключення до AWS (обліковий профіль, регіони, роль для AssumeRole), перелік активних модулів перевірок та налаштування для обчислення метрик. Конфігурація також визначає формат звіту, порогові значення для окремих індикаторів та вагові коефіцієнти, що впливають на розрахунок інтегрального показника SPS. Такий підхід дозволяє адаптувати роботу утиліти до різних сценаріїв: одноразового аудиту, регулярного сканування у CI/CD чи аналізу окремих ресурсів.

2. Ініціалізація провайдера та встановлення з'єднання з AWS

На основі отриманої конфігурації утиліта створює об'єкт провайдера - компонент, відповідальний за взаємодію з AWS через офіційний SDK. Цей модуль реалізує загальний інтерфейс ProviderAPI, що забезпечує абстракцію поверх конкретних хмарних платформ. У поточній версії реалізовано підтримку AWS, однак архітектура дозволяє без змін у логіці додати Azure чи GCP у майбутньому.

У межах цього етапу провайдер встановлює сесії boto3, визначає доступні регіони для збору даних, виконує автентифікацію через профіль або механізм AssumeRole і перевіряє мінімальний набір дозволів, необхідних для роботи інструмента.

3. Збір даних (Collector phase)

Після успішної ініціалізації активується модуль збору даних. Його завданням є отримання актуального стану ключових компонентів хмарного середовища: IAM-конфігурацій, S3-бакетів, мережевої інфраструктури, журналів CloudTrail, ключів KMS та інших сервісів. На цьому етапі враховуються:

- робота з багатьма регіонами;
- коректна обробка пагінації та великих результатів;
- ідентифікація ресурсів із застарілими або неконсистентними налаштуваннями;

- перетворення сирих даних AWS у уніфіковані структури.

Нормалізація даних має важливе значення: вона дозволяє забезпечити можливість застосування спільного набору правил до ресурсів різних типів, спрощує подальший аналіз та уможливлює масштабування інструмента.

4. Застосування правил (Rules Engine)

Після формування колекції ресурсів запускається механізм перевірок. Rules Engine послідовно або паралельно застосовує набір визначених правил, кожне з яких відповідає певній вимозі безпеки. Правила охоплюють широкий спектр аспектів - від IAM і шифрування до мережевої безпеки, конфігураційних відхилень та журналювання.

Кожне правило має чітко визначену структуру:
- опис вимоги (policy requirement);
- умови виконання та винятки;
- критерії оцінювання;
- рівень критичності;
- відповідність ключовим метрикам із розділу 2.

Результатом роботи правил є набір структурованих findings, що описують як успішні перевірки, так і виявлені порушення.

5. Обчислення метрик і інтегрального показника SPS

На основі результатів перевірок утиліта переходить до обчислення метрик ефективності безпеки. Модуль метрик агрегує дані відповідно до моделі, розробленої в попередньому розділі, і формує такі індикатори, як:
- Compliance Rate;
- Coverage Index;
- Automation Level;
- рівень залишкового ризику;
- інші категорійні показники.

Після цього розраховується інтегральний показник SPS - узагальнена оцінка стану безпеки, що враховує ваги окремих категорій ризиків і дозволяє порівнювати результати між різними середовищами.

6. Формування звіту та відображення результатів

На завершальному етапі утиліта формує інтегрований звіт, що включає:
- список знайдених порушень і попереджень;
- агреговані значення метрик;
- підсумковий SPS;

- рекомендації для виправлення виявлених недоліків.

Формат звіту визначається у конфігурації та може включати консольний вивід, JSON/YAML або структурований Markdown/HTML-документ.

Загальна схема роботи інструмента забезпечує відтворюваний, автоматизований і масштабований підхід до оцінювання безпеки AWS-акаунта. Вона реалізує методологію, розроблену в теоретичній частині роботи, і створює основу для практичного застосування формалізованих критеріїв та метрик.

4.2. Принципи проєктування

Проєктування утиліти для автоматизованого аналізу стану безпеки AWS-акаунта грунтується на низці архітектурних, методологічних та концептуальних принципів, які забезпечують масштабованість, надійність і можливість подальшого розвитку рішення. У цьому підрозділі описано ключові підходи, що визначили загальну логіку побудови інструмента.

1. Модульність та чітке розмежування відповідальностей

Основною вимогою при проєктуванні було розділення системи на окремі модулі, кожен з яких відповідає за свій етап обробки:

- модуль провайдера - за взаємодію з AWS;
- модуль збору даних - за агрегацію інформації про ресурси;
- rules engine - за аналіз і застосування політик;
- модуль метрик - за перетворення результатів перевірок у кількісні показники;
- модуль звітності - за подання результатів у зручній формі.

Таке розмежування дозволяє спростити обслуговування системи, впроваджувати нові функції без порушення існуючої логіки та зменшує ризик побічних ефектів при зміні окремих компонентів.

2. Незалежність від конкретного хмарного провайдера

Хоча поточна реалізація орієнтована на AWS, архітектура з самого початку була спроєктована як провайдер-незалежна. Це досягається шляхом:

- створення абстрактного інтерфейсу ProviderAPI;
- уніфікованого формату внутрішніх ресурсів;
- розподілу логіки на рівень “збір даних” і рівень “правил”;
- можливості додати Azure або GCP, реалізувавши відповідний підмодуль, не змінюючи rules engine чи metrics.

Такий підхід також демонструє, що система спирається не на конкретні сервіси, а на універсальні принципи хмарної безпеки, описані в розділі 3.

3. Принцип мінімальних привілеїв для самої утиліти

Оскільки застосунок працює безпосередньо з конфігурацією хмарного середовища, важливо, щоб він:
- не вимагав надмірних прав;
- працював лише з read-only дозволами;
- мав можливість використовувати AssumeRole з обмеженою політикою;
- не створював додаткових ризиків компрометації.

Тому при проєктуванні передбачено окремий IAM-профіль з мінімально необхідними дозволами, а утиліта не виконує жодних операцій зміни ресурсів - лише читає їхній стан.

4. Масштабованість та стійкість до збільшення обсягів даних

AWS-акаунт може містити тисячі ресурсів, розташованих у кількох регіонах. Тому утиліта повинна:
- коректно працювати з пагінацією API;
- підтримувати паралельну обробку незалежних наборів ресурсів;
- мати ефективний механізм кешування й нормалізації;
- витримувати збільшення кількості правил та обсягів метрик.

Стійкість до масштабування є критично важливою, оскільки надмірно “важкі” інструменти логічно не застосовуються в реальних DevSecOps-процесах.

5. Відтворюваність та детермінованість результатів

Однією з вимог до інструмента є можливість повторного запуску з однаковими результатами за умови незмінного стану середовища. Це забезпечується:
- чіткою нормалізацією даних;
- стандартизованою структурою правил;
- відсутністю випадкових або неформалізованих алгоритмів;
- можливістю фіксації версій правил у конфігурації.

Такі властивості особливо важливі для аудиту, аналітичних звітів та оцінювання ефективності змін у часі.

6. Прозорість та простежуваність (traceability)

Для будь-якого знайденого порушення повинно бути зрозуміло:
- яке саме правило його спричинило;
- які дані були використані;
- який критерій було порушено;
- як це порушення впливає на метрики та SPS.

Ця прозорість відрізняє інструмент від «чорних ящиків» типу комерційних CSPM-рішень і робить його придатним для навчання, досліджень і внутрішнього аудиту.

7. Підтримка стандартизованих форматів конфігурацій та звітності

Використання YAML-файлів для конфігурації та JSON/YAML/Markdown/HTML для звітів забезпечує:

- легку інтеграцію з CI/CD;
- використання інструмента у пайплайнах DevSecOps;
- можливість подальшого аналізу сторонніми системами;
- уніфікацію процедур запуску в команді або організації.

Таким чином утиліта природно вбудовується у сучасні робочі процеси.

4.3. Структура модулів системи

Архітектура розробленої утиліти має чітко окреслену модульну структуру, яка дозволяє незалежно розвивати окремі компоненти, розширювати функціональність і зберігати зрозумілий поділ відповідальностей. Завдяки цьому інструмент поєднує відтворюваність, масштабованість і простоту впровадження в реальні робочі процеси. Кожен модуль виконує певну роль у загальному циклі роботи системи, створюючи ланцюг взаємопов’язаних етапів, де результат одного стає підгрунтям для наступного.

Першим ключовим компонентом є модуль взаємодії з хмарним провайдером. У поточній реалізації він відповідає за роботу з AWS, використовуючи офіційний SDK для автентифікації, встановлення сесій та отримання даних із сервісів. Його роль полягає у забезпеченні стабільного, безпечного та відокремленого доступу до API провайдера. Від початку модуль проєктувався таким чином, щоб не прив’язувати утиліту виключно до AWS: завдяки абстракції на рівні інтерфейсу ProviderAPI ця частина може бути легко замінена або доповнена іншими реалізаціями для Azure чи GCP.

Наступним компонентом є модуль збору даних (collector), який забезпечує отримання актуального «знімка» стану хмарного середовища. Його головне завдання - зібрати відомості про IAM-конфігурації, S3-бакети, мережеві налаштування, журнали CloudTrail, ключі шифрування KMS та інші критичні ресурси. Collector не містить логіки безпеки; він виконує виключно функцію агрегації даних та їх нормалізації. Це важливо, оскільки уніфікована структура ресурсів дозволяє застосовувати однакові правила до різних типів об’єктів і забезпечує стабільність роботи всієї системи незалежно від різнорідності початкових даних.

Центральним елементом архітектури є модуль правил перевірок - rules engine. Саме тут відбувається застосування визначених вимог безпеки до зібраних даних. Кожне правило формалізує певний аспект безпечної

конфігурації: наприклад, недопустимість відкритих S3-бакетів, використання надмірних IAM-привілеїв або відсутність CloudTrail у всіх регіонах. Важливо, що правила не залежать одне від одного, можуть бути ввімкнені чи вимкнені через конфігураційний файл та легко оновлюються. Таке проєктне рішення забезпечує гнучкість і дозволяє масштабувати систему у відповідь на динаміку рекомендацій або стандартів безпеки.

Обробка результатів перевірок відбувається у модулі метрик, який реалізує методологію оцінювання ефективності, описану у другому розділі роботи. Саме тут зібрані findings перетворюються на кількісні індикатори - від рівня відповідності (Compliance Rate) до інтегрального показника SPS, що узагальнює загальний стан хмарної інфраструктури. Модуль метрик враховує значущість різних категорій порушень, дозволяє налаштовувати вагові коефіцієнти та формує підгрунтя для порівняння результатів між різними аудитами або хмарними акаунтами.

Окремим елементом системи є модуль формування звітів, який відповідає за подання результатів аналізу у структурованому та зрозумілому вигляді. Він підтримує кілька форматів - консольний, JSON/YAML для автоматизації або Markdown/HTML для включення в документацію чи внутрішні звіти. Звіти містять як детальну інформацію про кожне виявлене порушення, так і зведені метрики з їх інтерпретацією, що робить результат аналізу придатним як для технічних спеціалістів, так і для менеджерів із безпеки чи IT-керівників.

Завершальним компонентом є командний інтерфейс, який забезпечує запуск утиліти, вибір конфігураційного файлу, активацію окремих перевірок і визначення формату звіту. Хоча цей модуль не є основним з точки зору аналітичної логіки, він визначає зручність використання інструмента, уможливлює його інтеграцію в CI/CD-процеси та робить утиліту придатною для автоматизованого або регулярного застосування.

Таким чином, модульна структура утиліти формує цілісну, відтворювану та гнучку архітектуру, яка легко масштабується та адаптується під нові вимоги. Вона дозволяє незалежно розвивати логіку збору даних, аналізу, обчислення метрик та формування звітів, що робить розроблене рішення придатним для подальшого розвиту і для практичного застосування як у наукових цілях, так і в середовищах реальних організацій.

4.4. Механізм збору даних

Механізм збору даних є фундаментом усієї утиліти, оскільки саме від повноти і якості одержаної інформації залежить коректність подальшого аналізу, результатів перевірок та розрахунку метрик ефективності. Збір даних у хмарному середовищі має низку особливостей, які відрізняють його від традиційного аудиту локальних інфраструктур і вимагають специфічних підходів до проєктування. AWS, як і інші хмарні платформи, є розподіленою системою з великою кількістю сервісів, регіонів, ресурсів та API-інтерфейсів, що потребує узгодженого й системного процесу агрегації даних.

Першою вимогою є забезпечення консистентності зібраних даних. Оскільки хмарні ресурси можуть змінюватися у реальному часі - видалятися, модифікуватися, масштабуватися - механізм збору повинен працювати так, щоб отриманий «знімок» стану інфраструктури був логічно цілісним. Це означає, що всі дані, які використовуються при подальшому аналізі, повинні відповідати одному часовому контексту. Хоча абсолютна синхронність у розподілених системах недосяжна, утиліта повинна мінімізувати часові розриви між запитами, а також уникати дублювання або часткових результатів.

Другою важливою вимогою є здатність коректно працювати з великими обсягами даних. AWS API повертають інформацію порціями, залежно від сервісу та регіону, тому система має враховувати пагінацію, оптимізувати кількість запитів та гарантувати, що всі ресурси будуть охоплені. Особливо це важливо для таких сервісів, як IAM (з великою кількістю ролей і політик), S3 (де число бакетів може бути суттєвим), чи EC2/VPC (які містять значні конфігураційні структури).

Окремого підходу вимагає робота з багатьма регіонами, оскільки AWS-функціональність не є централізованою. Більшість сервісів конфігурується у межах конкретного регіону, і критично важливі дані щодо безпеки (наприклад, журнали CloudTrail або налаштування Security Groups) можуть бути розподілені в декількох областях. Механізм збору даних повинен вміти автоматично визначати доступні регіони, звертатися до кожного з них окремо та гарантувати, що результати будуть агреговані правильно та без втрати контексту.

Ще одним аспектом є мінімація привілеїв. Збір даних повинен відбуватися з використанням ролі або профілю, який має виключно read-only доступ до необхідних сервісів. Невиправдано широкі дозволи або доступ до операцій запису суперечили б принципам безпеки, описаним у розділі 3, і могли б створити додаткові ризики для організації. Тому утиліта спирається на окремий IAM-профіль із обмеженим набором дозволів, що також підвищує прозорість і відтворюваність результатів.

Важливою вимогою є також стійкість до мережевих помилок і rate limits. AWS може тимчасово відхиляти запити через перевищення лімітів або інтенсивне використання API. Тому механізм збору повинен включати обробку повторних спроб, експоненційні затримки та детальне логування, щоб уникнути часткового або неповного отримання інформації.

Нарешті, критичним елементом є нормалізація даних. Різні сервіси AWS повертають інформацію в різних форматах, і без попередньої стандартизації ця інформація не може бути ефективно проаналізована. Тому після збору даних утиліта перетворює їх у внутрішні уніфіковані моделі, у яких чітко структуруються властивості ресурсів, пов’язані доступи, політики, стани безпеки та інші атрибути. Нормалізація забезпечує можливість застосування єдиного набору правил незалежно від того, який сервіс було проаналізовано.

Таким чином, загальні вимоги до збору даних формують базу для надійної та відтворюваної роботи всієї утиліти. Вони забезпечують повноту, консистентність та точність інформації, що у свою чергу є необхідними умовами для коректної роботи правила перевірок, обчислення метрик та формування

інтегрального показника SPS. Саме ці вимоги відрізняють інструмент від спрощених або ручних методів аудиту та роблять його придатним для системного аналізу хмарної інфраструктури.

Ефективність роботи утиліти значною мірою визначається тим, наскільки повно і точно вона здатна відобразити реальний стан хмарного середовища. З огляду на це, механізм збору даних має охоплювати ключові компоненти AWS-інфраструктури, які мають безпосередній вплив на безпеку: ідентифікацію та доступ, конфігурацію сховищ, мережеву модель, журналювання подій, а також засоби шифрування. Кожен із цих напрямів містить у собі низку важливих параметрів, що дозволяють сформувати вичерпну картину середовища та визначити відповідність ресурсів політикам безпеки.

Однією з найважливіших категорій є дані про систему ідентифікації та доступу (IAM). Утиліта збирає інформацію про ролі, користувачів, групи та призначені їм політики, включно з inline-політиками та прив’язаними managed-policy. Також враховуються дані про наявність застарілих або невикористовуваних ключів доступу, рівень деталізації привілеїв, використання wildcard-дозволів і відповідність принципам найменших привілеїв. Ця категорія є центральною для оцінювання ризиків, оскільки саме неправильна конфігурація доступів часто стає причиною інцидентів у хмарних середовищах.

Не менш важливим є збір даних про сховища та сервіси збереження інформації, передусім S3. У цьому контексті утиліта отримує відомості про налаштування політик доступу, списки контролю доступів (ACL), конфігурацію шифрування, статус блокування публічного доступу та пов’язані механізми логування. Враховуючи, що S3 є однією з найчастіших точок витоку інформації, ця частина аналізу має значну вагу у загальному оцінюванні безпеки.

Окрему роль відіграє інформація про мережеву інфраструктуру, зокрема Virtual Private Cloud (VPC) та пов’язані з ним компоненти. Утиліта аналізує параметри Security Groups, маршрутизацію, список мережевих ACL, наявність приватних або публічних підмереж та використання приватних endpoint’ів для доступу до хмарних сервісів. Мережева конфігурація показує, наскільки середовище ізольоване від публічного доступу та чи відповідає воно принципам Zero Trust.

Ще одним важливим джерелом є журнали та налаштування CloudTrail, які відображають історію взаємодії з API AWS. Механізм збору даних отримує інформацію про те, чи увімкнено CloudTrail у всіх регіонах, чи ведеться журналювання управлінських подій, чи зберігаються логи у зашифрованому вигляді, а також як організовано їх збереження у S3. Ця інформація критично важлива для оцінювання спроможності організації виявляти інциденти та здійснювати аудит дій.

Важливу роль у забезпеченні безпеки відіграють також ключі шифрування та засоби Key Management Service (KMS). Утиліта збирає дані про наявні ключі, політики доступу до них, статус автоматичної ротації, а також виявляє так звані “ad-hoc” ключі, які використовуються у неконтрольованих сценаріях. Аналіз цієї інформації дозволяє оцінити, наскільки правильно побудовано процес управління секретами.

За потреби утиліта може збирати дані з додаткових сервісів, таких як RDS, Lambda або EBS, однак у базовій версії акцент робиться на тих компонентах, які мають найбільший вплив на загальний рівень безпеки середовища.

Усі зібрані дані після цього проходять етап нормалізації, завдяки чому різнорідні структури-IAM-ролі, об’єктні бакети чи мережеві політики-перетворюються на уніфіковані внутрішні моделі, придатні для подальшого застосування правил.

Таким чином, утиліта охоплює ключові частини хмарної інфраструктури, забезпечуючи повне та системне представлення конфігураційного стану середовища. Це дозволяє формувати об’єктивні результати перевірок та коректно обчислювати метрики, що описують ефективність реалізованих засобів безпеки.

4.5. Набір правил перевірок

Ефективність утиліти визначається не лише якістю зібраних даних, але й здатністю правильно інтерпретувати їх з точки зору безпеки. Саме тому центральним елементом системи є набір правил перевірок, або rules engine, який виконує функцію аналітичного ядра. Цей модуль перетворює фактичний стан ресурсів AWS у структуровані висновки про відповідність або невідповідність вимогам безпеки, окресленим у попередніх розділах роботи.

В основі rules engine лежить ідея чіткої формалізації кожної вимоги безпеки у вигляді окремого незалежного правила. Кожне правило описує конкретний аспект конфігурації - наприклад, чи є S3-бакет публічним, чи містить IAM-політика надмірні дозволи, чи налаштовано CloudTrail у всіх регіонах, чи проводиться ротація ключів KMS, чи відсутні відкриті порти у Security Groups. Такий підхід дозволяє створити прозору і розширювану систему, у якій додавання нових правил не порушує роботу існуючих, а зміна логіки виконання окремих перевірок не впливає на загальну архітектуру.

Правила працюють поверх нормалізованих даних, отриманих collector-модулем, що значно підвищує надійність і передбачуваність їх виконання. Завдяки цьому правила можуть бути універсальними - вони не залежать від того, з якого саме сервісу походять дані, а спираються виключно на внутрішні моделі ресурсів. Це дозволяє масштабувати систему на інші хмарні платформи або нові сервіси AWS без заміни основної логіки.

Rules engine також забезпечує єдину модель виведення результатів. Кожне правило повертає формалізований результат перевірки, що містить статус, опис проблеми, перелік задіяних ресурсів та рекомендацію щодо усунення недоліків. Такий підхід уможливлює агрегацію результатів у метрики, які будуть розраховані на наступному етапі, та забезпечує зв’язок між рекомендаціями розділу 3 і практичним оцінюванням середовища.

4.5.1. Концепція правил

Концепція правил є ключовою методологічною основою для всієї системи автоматизованого аналізу. У рамках утиліти кожне правило розглядається як окрема формалізована вимога безпеки, яка задає чіткі критерії того, що вважається коректною або некоректною конфігурацією. Це дозволяє інтерпретувати стан хмарного середовища не інтуїтивно, а у спосіб, який спирається на стандарти, рекомендації та принципи, викладені у теоретичних розділах.

Правило, у найзагальнішому сенсі, складається з трьох концептуальних частин. Перша - це опис політики або вимоги, яку має виконувати ресурс. Часто такий опис грунтується на стандартах безпеки (наприклад, CIS AWS Foundations Benchmark, NIST SP 800-53, CSA CCM) або на практиках, сформульованих у розділі 3. Друга частина стосується алгоритму перевірки: правило визначає набір умов, за яких ресурс вважається таким, що відповідає вимозі. Це може включати аналіз атрибутів ресурсу, перевірку зв’язків між ним та іншими ресурсами, а також визначення винятків, що пояснюють допустимі відхилення від стандартного сценарію. Третя частина - це інтерпретація результату: правило повинно не лише виявити порушення, але й надати його категоризацію, описати його наслідки та сформулювати рекомендацію для усунення.

Важливою властивістю такої концепції є незалежність правил одне від одного. У межах системи немає ієрархії між правилами чи залежностей, які б впливали на порядок їх виконання. Це забезпечує стабільність, простоту масштабування та можливість тонкого налаштування набору перевірок через конфігураційний файл. Користувач може вмикати або вимикати окремі правила, не змінюючи загального алгоритму роботи утиліти.

Ще однією ключовою характеристикою є прозорість. Кожне правило має бути описане достатньо чітко, щоб користувач міг зрозуміти, чому саме конфігурація була позначена як проблемна. Ця вимога особливо важлива у контексті магістерської роботи, оскільки підкреслює науковий характер підходу: інструмент не приховує логіку аналізу, а навпаки, робить її доступною для інтерпретації, навчання та вдосконалення.

Таким чином, концепція правил визначає не лише структуру аналітичного блоку утиліти, але й загальну методологічну базу автоматизованого оцінювання хмарної безпеки. Вона дозволяє забезпечити узгодженість між теоретичними положеннями та практичною реалізацією, роблячи систему придатною як для дослідницьких цілей, так і для застосування в реальних інфраструктурних середовищах.

4.5.2. Джерела формування правил

Правила, що реалізовані у розробленій утиліті, не є випадковим або довільним набором перевірок. Вони грунтуються на комплексі авторитетних стандартів, рекомендацій та практик, які визначають сучасні підходи до забезпечення хмарної безпеки. Формування кожного правила спирається на офіційні джерела, що задають вимоги до конфігурації сервісів, організації доступу, використання шифрування, журналювання та інших ключових аспектів.

Це дозволяє забезпечити не лише технічну коректність перевірок, а й методологічну обгрунтованість, що є критично важливим у контексті дослідницької роботи.

Основою для розробки правил стали міжнародні стандарти та рекомендації, які визнаються академічною спільнотою та широким колом професійних організацій. Перш за все, це CIS AWS Foundations Benchmark, який визначає конкретні вимоги щодо налаштування CloudTrail, IAM, S3, EC2 та інших сервісів. Даний стандарт є де-факто базовим документом для оцінювання стану безпеки AWS-акаунтів і широко використовується як у внутрішніх аудитах, так і у комерційних CSPM-платформах.

Важливу роль відіграє і сімейство рекомендацій NIST, зокрема документи SP 800-53 та SP 800-207. Перший описує систему контролів інформаційної безпеки, що може бути застосована у хмарних середовищах, а другий є фундаментом концепції Zero Trust, яка визначає принципи ізоляції, суворої автентифікації, сегментації мережі та мінімізації довіри між компонентами системи. Ці положення лягли в основу правил, що стосуються мережевої безпеки, контролю привілеїв та політик доступу.

Джерелом додаткових вимог до конфігурації хмарних сервісів стала Cloud Controls Matrix (CCM), розроблена Cloud Security Alliance (CSA). Ця матриця структурує вимоги у вигляді контрольних доменів і дозволяє пов’язати перевірки не лише з технічними аспектами, а й з управлінськими процесами. Для нашої утиліти вона особливо важлива тим, що дозволяє мапувати перевірки до високорівневих принципів безпеки, таких як управління доступом, захист даних, журналювання, моніторинг та відповідність політикам.

Не менш важливими є рекомендації AWS Well-Architected Framework (Security Pillar). Цей документ містить набір практичних порад щодо побудови безпечних хмарних архітектур, які є більш прикладними, ніж формальні стандарти. Вони безпосередньо вплинули на правила, пов’язані з мережею, IAM, KMS та CloudTrail, а також надали чіткі орієнтири щодо того, як відрізнити “допустиму конфігурацію” від “небезпечної”.

Окрім офіційних стандартів, частину правил було сформовано на основі рекомендацій, описаних у третьому розділі цієї роботи. Ці рекомендації базуються на аналізі актуальних загроз, статистики інцидентів та властивих хмарним середовищам вразливостей. На відміну від міжнародних стандартів, вони мають прикладний характер і адаптовані до практики використання AWS у реальних інфраструктурах. Утиліта таким чином є не лише реалізацією зовнішніх вимог, але й практичним втіленням методології, розробленої в рамках дослідження.

Для зручності нижче наведено ключові документальні джерела, на які спираються правила:

– CIS AWS Foundations Benchmark (v1.4–1.5) - вимоги щодо CloudTrail, IAM, S3, моніторингу, шифрування.

- NIST SP 800-53 - контроли AC, AU, SC, CM, що визначають правила управління доступом, журналювання, конфігураційного контролю та шифрування.
- NIST SP 800-207 (Zero Trust Architecture) - принципи ізоляції, сегментації та мінімізації довіри.
- CSA Cloud Controls Matrix (CCM) - мапінг перевірок до доменів безпеки, таких як IAM, Data Security, Application Security, Threat Monitoring.
- AWS Well-Architected Security Pillar - практичні рекомендації для побудови безпечної та масштабованої архітектури.

Таким чином, правила утиліти мають комплексну та науково обгрунтовану основу, яка поєднує вимоги міжнародних стандартів, галузеві рекомендації та результати власного дослідження. Це забезпечує високу якість, репрезентативність і практичну цінність результатів, отриманих за допомогою інструмента.

4.5.3. Категоризація правил

Набір правил, реалізованих у утиліті, охоплює широкий спектр аспектів хмарної безпеки. Для забезпечення системності та прозорості аналізу правила були згруповані у тематичні категорії, відповідно до яких організується і процес оцінювання, і формування результатів. Така категоризація відображає структурну логіку стандартів, на які спирається інструмент, а також дозволяє узгоджувати практичні перевірки з визначеними у другому розділі метриками. Вона також забезпечує узгодженість між правилами та критичністю ризиків, характерних для різних частин хмарної інфраструктури.

У найзагальнішому сенсі категорії правил відповідають ключовим областям безпеки, властивим хмарним середовищам. Першою та однією з найважливіших є категорія управління доступом (Identity and Access Management). Правила в цій групі аналізують пов’язані з IAM ресурси: користувачів, ролі, політики, ключі доступу та механізми автентифікації. Основний акцент робиться на виявленні надмірних дозволів, використанні wildcard-політик, відсутності двофакторної автентифікації, існуванні старих або невикористаних ключів та загальних відхилень від принципу найменших привілеїв.

До другої важливої групи належать правила, що оцінюють безпеку збереження даних, насамперед конфігурації S3-бакетів. У цій категорії розглядаються налаштування політик доступу, статус публічного доступу, шифрування даних у спокої, використання механізмів versioning, а також відповідність схем доступу внутрішнім політикам. Враховуючи часту появу інцидентів, пов’язаних із неправильним налаштуванням S3, ця категорія має значний вплив на фінальні метрики.

Окрема група правил стосується мережевої безпеки. Вона охоплює аналіз Security Groups, списків контролю доступу VPC, маршрутних таблиць, інтернет-шлюзів, NAT, приватних та публічних підмереж. Особлива увага приділяється

виявленню відкритих портів, порушень сегментації, надмірної експозиції ресурсів у публічний інтернет та відсутності захисних механізмів для контролю мережевого трафіку.

Четверта категорія охоплює правила, пов’язані з журналюванням і моніторингом. Її мета - оцінити, чи організація має можливість вчасно виявляти інциденти, розслідувати їх та встановлювати першопричини. Перевірки включають статус CloudTrail, наявність журналювання в усіх регіонах, коректність зберігання логів, використання шифрування та інтеграцію з іншими сервісами моніторингу.

П’ята категорія пов’язана з шифруванням та управлінням ключами (KMS). У межах цієї групи аналізуються налаштування ключів шифрування, правильність політик, використання автоматичної ротації, а також випадки надмірної відкритості ключових політик, що може поставити під загрозу конфіденційність даних.

До окремої групи належать правила, спрямовані на виявлення конфігураційного дрейфу - ситуацій, у яких ресурси відхиляються від нормальної або очікуваної конфігурації. Це включає розбіжності між фактичними налаштуваннями та політиками, визначеними у IaC, або типові моделі помилок, що виникають унаслідок ручних змін. Такий напрямок має особливе значення у DevSecOps-середовищах, де інфраструктура є динамічною.

Для більшої наочності нижче подано приклади категорій у структурованому вигляді:

- Identity & Access Management: надмірні дозволи, відсутність MFA, застарілі ключі, непрозорі політики.
- Storage Security: публічні бакети, відсутність шифрування, слабкі ACL, неправильні bucket policies.
- Network Security: відкриті порти, помилки сегментації, надмірна експозиція ресурсів, слабкі правила SG.
- Logging & Monitoring: неактивний CloudTrail, відсутність журналювання у регіонах, незахищене зберігання логів.
- Encryption & Key Management: відсутність ротації, неправильні key policies, використання неавторизованих ключів.
- Configurational Drift: неконтрольовані відхилення від політик, розбіжності із IaC, випадкові ручні зміни.

Така систематизація дозволяє не лише організувати сам процес перевірок, але й безпосередньо впливає на формування метрик. Наприклад, Compliance Rate може обчислюватися окремо для кожної категорії, а індекс покриття - враховувати наявність правил у кожній з них. Це підсилює аналітичні можливості утиліти та надає більш детальну картину ризиків, властивих конкретному акаунту.

4.6. Обчислення метрик та інтегрального показника SPS

Метрики, що обчислюються утилітою, мають безпосередній зв’язок із категоріями правил описаних раніше. Для кожного правила система формує оцінку «пройдено / не пройдено», визначає рівень критичності та категорію, до якої належить виявлене порушення. Ця інформація пізніше агрегується у кількісні індикатори, які відображають якість конфігурації середовища в цілому.

Нижче наведено метрики, які утиліта формує за результатами аналізу. Вони є формалізованим продовженням критеріїв, представлених у другому розділі.

Compliance Rate (CR)

Це базовий показник, який відображає частку правил, що були виконані успішно. CR показує:

- загальний рівень відповідності вимогам,
- тенденції у змінах стану конфігурації,
- непрямий рівень організаційної дисципліни у сфері безпеки.

Утиліта обчислює CR як у загальному вигляді, так і для окремих категорій (IAM, S3, мережа, журналювання тощо).

Coverage Index (CI)

Показник відображає, наскільки повно середовище охоплене перевірками. Наприклад, якщо організація не використовує CloudTrail у деяких регіонах, coverage для відповідної категорії буде нижчим. CI дає змогу зрозуміти, наскільки результати аналізу можна вважати репрезентативними.

Failure Density (FD)

Ця метрика показує концентрацію порушень у певних категоріях або сервісах. Вона дозволяє виявити області, де проблем накопичується найбільше (наприклад, IAM чи мережевий рівень), та визначити, де слід зосередити пріоритети безпеки.

Critical Findings Ratio (CFR)

Метрика показує частку порушень, які мають високий рівень критичності. Враховуючи, що не всі findings є однаково небезпечними, CFR є важливою складовою SPS. Вона сигналізує, чи є середовище критично вразливим, навіть якщо загальний рівень compliance може бути високим.

Resource Exposure Index (REI)

Індекс визначає рівень експозиції ресурсів до зовнішнього світу. До уваги беруться:
– відкриті порти,
– публічні S3-бакети,
– некоректно сконфігуровані Security Groups,
– відсутність сегментації.

REI особливо важливий у контексті Zero Trust.

Encryption Compliance Level (ECL)

Метрика показує частку ресурсів, які використовують шифрування у спокої або під час передачі даних. Вона базується на правилах, що стосуються S3, EBS, RDS, KMS і мережевих протоколів.

Logging & Monitoring Readiness (LMR)

Показує, наскільки середовище готове до виявлення інцидентів. Враховує:
– статус CloudTrail,
– охоплення регіонів,
– стан Flow Logs,
– налаштування retention.

У цій метриці поєднується технічна конфігурація та організаційна готовність

Key Management Maturity (KMM)

Окремий показник оцінює:
– стан політик KMS,
– рівень контролю доступу до ключів,
– наявність автоматичної ротації,
– випадки надмірно відкритих key policy.

KMM є важливим індикатором безпеки даних.

Drift Detection Index (DDI)
Відображає частку ресурсів, конфігурація яких відхиляється від стандартної або очікуваної. Враховуються:
– відхилення IAM-політик від рекомендованих шаблонів,
– ручні зміни в SG,
– невідповідності між CloudTrail і політиками журналювання.

Інтегральний показник SPS

SPS поєднує всі попередні метрики, використовуючи вагову модель. Його значення:

- дає цілісну оцінку безпеки,
- дозволяє порівнювати різні акаунти,
- підходить для executive-level звітів,
- є індикатором ефективності впроваджених рекомендацій.

SPS опирається на принципи, визначені у розділі 2, і є ключовим результатом утиліти.

4.7. Формування звіту

Формування звіту є завершальним етапом роботи утиліти і відіграє ключову роль у трансформації технічних результатів перевірок у форму, придатну для інтерпретації користувачами різного рівня - від інженерів до керівників підрозділів з інформаційної безпеки. Якщо попередні етапи роботи утиліти були зосереджені на зборі даних та аналітичній обробці, то саме модуль формування звітів забезпечує доступність результатів, їх осмислення та практичну застосовність.

Звіт виконує одразу кілька функцій. По-перше, він документує фактичний стан хмарної інфраструктури на момент запуску утиліти, фіксуючи як виявлені проблеми, так і успішні перевірки. По-друге, він структурує знайдені порушення за категоріями, критичністю та впливом на безпеку, що дозволяє швидко визначити пріоритети та оцінити загальний рівень ризику. По-третє, звіт включає розраховані метрики та інтегральний показник SPS, які надають можливість сформувати цілісне уявлення про зрілість процесів безпеки та їх відповідність політикам і стандартам.

Зміст і формат звіту визначаються параметрами в конфігураційному файлі. Це дозволяє адаптувати звіт до різних сценаріїв використання. У середовищах DevSecOps зазвичай використовується компактний текстовий формат або JSON, придатний для інтеграції з CI/CD-пайплайнами. Для аналітичних або аудиторських потреб утиліта може формувати структурований Markdown чи HTML-документ, який підходить для включення у звіти безпеки, технічну документацію або презентації.

Незалежно від формату, структура звіту є сталою та складається з кількох логічних частин. Спочатку подається загальна інформація про запуск: дата і час аналізу, цільовий акаунт, регіони, активні правила та параметри конфігурації. Це забезпечує можливість відтворити аналіз у майбутньому та дозволяє зіставляти результати між різними аудитами.

Далі подається основний розділ, присвячений знайденим порушенням. Він містить детальний опис кожної проблеми, включно з її типом, рівнем критичності, задіяними ресурсами та поясненням, чому саме було визнано проблему. Особливістю підходу є те, що кожен запис супроводжується рекомендацією - конкретною дією, яка дозволяє усунути виявлене порушення.

Це робить звіт не лише діагностичним, але й практично орієнтованим інструментом.

На наступному етапі звіт містить агреговані показники. До них належать:
- частка успішних перевірок (Compliance Rate),
- індекс охоплення середовища (Coverage Index),
- показники критичності виявлених проблем,
- індекс експозиції ресурсів,
- рівень захищеності засобів шифрування,
- зрілість механізмів журналювання і моніторингу,
- інтегральний показник SPS.

Ці метрики подаються як числові значення з коротким аналітичним коментарем, що пояснює їх значення для стану безпеки середовища. Такий підхід дозволяє читачеві не лише бачити цифри, але й розуміти, що вони означають у контексті конкретної інфраструктури.

У розширених форматах (Markdown і HTML) звіт може також містити візуалізації: графіки розподілу проблем за категоріями, теплові карти, порівняння між регіонами або історію змін метрик за кілька запусків. Це підвищує інформативність звіту і робить його зручним для презентацій або стратегічного планування заходів безпеки.

Завершальна частина звіту містить підсумкові висновки щодо стану хмарного середовища, а також пропозиції щодо першочергових дій. Завдяки узгодженості з рекомендаціями, сформульованими в третьому розділі, звіт стає інструментом практичного застосування теоретичних результатів дослідження.

Таким чином, модуль формування звітів виконує функцію мосту між технічною аналітикою та її інтерпретацією. Він забезпечує прозорість, відтворюваність і практичну цінність усього процесу аудиту, перетворюючи незліченні результати правил і метрик у структуроване уявлення про стан безпеки AWS-акаунта.

4.8 Результати роботи утиліти

Покроковий алгоритм роботи додатку:
1. Програма читає файл конфігурацій(додаток А)
2. Відбувається повний збір інформації з переданих джерел(аккаунтів)
3. Всі отримані ресурси проходять через усі доступні правила перевірки
4. Знайдені вразливості отримують певну оцінку в залежності від рівня небезпеки знайденої вразливості
5. Формується загальний звіт на основі знайдених вразливостей, їх рівня небезпеки і детального описання проблеми

6. Формується другий звіт з загальними цифрами для подальшого аналізу іншими інструментами, сюди складаються агреговані результати сканування

В результаті виконання утиліти генерується повноцінний звіт в 2 форматах markdown і json файлів 2 видів: з усіма знайденими вразливостями(додаток Б показує частину звіту) та загальним звітом по кількості знайдених вразливостей по сервісам(додаток В).

Висновок

Розділ 4 був присвячений практичній реалізації підходів до забезпечення безпеки хмарних інфраструктур, розроблених та обґрунтованих у попередніх частинах роботи. У цьому розділі здійснено перехід від теоретичних рекомендацій і формалізованих критеріїв до створення конкретного технічного рішення — автоматизованої утиліти для перевірки стану безпеки AWS-акаунта. Такий підхід дозволив не лише продемонструвати застосовність теоретичних положень, а й перевірити їхню ефективність у реальних умовах експлуатації хмарних сервісів.

У процесі розробки утиліти було сформовано архітектурну модель, орієнтовану на масштабованість, модульність і можливість подальшого розширення під мультихмарні середовища. Система збору даних реалізована з урахуванням вимог до мінімальної інвазивності та максимальної точності, що забезпечує отримання структурованих відомостей про основні компоненти інфраструктури: IAM, S3, VPC, KMS, CloudTrail та додаткові сервіси. Запропонована логіка обробки інформації гарантує відокремлення процесу збору від процесу аналізу, що сприяє підвищенню надійності, прозорості та відтворюваності результатів.

Особливу увагу приділено формуванню набору правил оцінювання, які узгоджуються з міжнародними стандартами безпеки (CIS Benchmarks, NIST, ISO/IEC 27001, CSA CCM) і водночас враховують практичні рекомендації, викладені у попередніх розділах дослідження. Кожне правило описує конкретну вразливість або відхилення від рекомендованих налаштувань та включає логіку визначення, ступінь критичності та можливі шляхи усунення. Завдяки цьому утиліта не просто виявляє проблеми, а й формує основу для систематичного вдосконалення безпеки.

Механізм формування звітів забезпечує адаптивність результатів до різних сценаріїв використання. Підтримка текстових, структурованих і розширених візуалізованих форматів дозволяє як інтегрувати утиліту в автоматизовані DevSecOps-процеси, так і використовувати її для підготовки аналітичних документів, аудитів або внутрішніх перевірок. Обчислення метрик та інтегрального показника безпеки (Security Posture Score), розроблених у розділі 2, забезпечує можливість кількісної оцінки стану середовища та порівняння результатів між окремими запусками чи різними акаунтами.

Таким чином, розділ 4 підтверджує, що запропоновані теоретичні підходи можуть бути реалізовані у вигляді повноцінного інструменту, який здатен

автономно виявляти критичні проблеми безпеки та створювати обґрунтовані й прозорі аналітичні звіти. Розроблена утиліта є не лише демонстраційним прототипом, а й практично корисним рішенням, яке вже на поточному етапі може застосовуватися в робочих середовищах. Її модульність і відкритість роблять інструмент придатним для подальшого розширення відповідно до потреб мультихмарних архітектур, що створює перспективи розвитку у напрямі автоматичного виправлення виявлених конфігураційних помилок, інтеграції машинного навчання для аналізу аномалій та розширення системи метрик.

Загалом, результати цього розділу демонструють практичну цінність проведеного дослідження та підтверджують, що автоматизація перевірок конфігураційної безпеки може суттєво знизити ризики, характерні для сучасних хмарних середовищ. Створений інструмент виступає логічним підсумком усього дослідження та засвідчує можливість успішної інтеграції теоретичних рекомендацій у реальні інженерні рішення.

ВИСНОВКИ

Проведене магістерське дослідження було спрямоване на всебічне вивчення методів захисту інформації в хмарних середовищах та розроблення системи рекомендацій і інструментальних засобів, здатних підвищити рівень безпеки сучасних хмарних інфраструктур. Робота поєднує теоретичний аналіз, огляд сучасних загроз, формалізацію системи критеріїв оцінювання та практичне створення програмної утиліти, що дозволяє застосувати розроблену методологію на реальних прикладах. Така комплексність дозволяє розглядати отримані результати не лише як окремі наукові положення, а як завершену, логічно узгоджену концепцію.

У першому розділі було обгрунтовано важливість теми дослідження, визначено ключові характеристики хмарних сервісів і проаналізовано, чому саме специфіка хмарних моделей формує унікальні ризики безпеки. Хмарні середовища надають широкий спектр можливостей — масштабованість, зменшення операційних витрат, доступність ресурсів — але одночасно створюють нові вразливості, пов’язані з багатокористувацькістю, автоматизованим управлінням ресурсами, мережею з розподіленою архітектурою та залежністю від механізмів доступу. Статистичні дані CSA, ENISA, IBM та Verizon підтвердили, що більшість інцидентів у хмарі є наслідком неправильних конфігурацій, надмірних привілеїв або недостатнього журналювання, а не технічних вразливостей самих провайдерів. Таким чином, уже на рівні першого розділу було показано, що основним джерелом ризиків є людський чинник і слабкість процесів контролю, що визначило напрям подальшого дослідження.

Другий розділ сформував методологічний фундамент роботи — систему критеріїв і метрик, здатних об’єктивно оцінювати стан безпеки хмарної інфраструктури. На відміну від традиційного підходу, де вимоги безпеки мають описовий характер, у роботі запропоновано формалізовану модель оцінювання. Було виділено загальні критерії (відповідність, повнота, узгодженість, своєчасність), економічні критерії (витрати на контроль, вартість інцидентів, оптимальність конфігурації), а також технічні критерії, що стосуються власне стану хмарного середовища. На цій основі сформовано набір метрик — Compliance Rate, Coverage Index, Failure Density, Resource Exposure Index, Encryption Compliance Level, Logging Readiness та інтегральний показник SPS. Саме цей розділ дав змогу перейти від загального опису проблем до вимірюваного підходу, що критично важливо для автоматизації безпеки в хмарі.

У третьому розділі було розроблено комплекс рекомендацій, які охоплюють ключові галузі хмарної безпеки: управління доступом, сегментація мережі, захист даних, шифрування, журналювання, управління ключами та контроль конфігурацій. Важливо, що рекомендації не дублюють сформульовані стандарти, а узгоджують їх у контексті потреб реальних хмарних середовищ, виділяючи саме ті заходи, що мають найбільший вплив на зменшення ймовірності інцидентів. У цьому розділі було підкреслено критичну роль автоматизації, безперервного моніторингу та відмови від ручних змін у

конфігурації — положення, які пізніше були покладені в основу програмної реалізації.

Четвертий розділ сформував практичну частину дослідження — розробку утиліти для аналізу стану AWS-акаунта. Детальний опис архітектури, принципів проєктування та структури модулів показав, що система створена з урахуванням вимог до масштабованості, розширюваності та відтворюваності. Утиліта реалізує повний цикл автоматичного аналізу: від збирання даних про IAM, S3, VPC, CloudTrail та KMS до застосування набору правил, обчислення метрик і формування багатоформатних звітів. Rules engine, побудований на основі CIS Benchmarks, NIST 800-53, CSA CCM та рекомендацій, сформульованих у попередніх розділах, забезпечує ідентифікацію критичних відхилень у конфігурації та дозволяє безпосередньо оцінювати ефективність заходів безпеки.

Систему метрик, описану у другому розділі, було повністю реалізовано у програмному продукті. Інтегральний показник SPS дає змогу оцінити загальний рівень безпеки в числовому вигляді, що відкриває шлях до побудови історичних трендів, порівняння різних акаунтів у межах організації або використання утиліти у DevSecOps-процесах. Формування звітів у форматах JSON, YAML, Markdown та HTML робить інструмент придатним як для інженерів, так і для аналітиків, аудиторів чи керівників підрозділів.

Таким чином, дослідження логічно поєднало теоретичну частину з практичною та підтвердило, що ефективність захисту в хмарних інфраструктурах може бути значно підвищена завдяки переходу від фрагментарних перевірок до системного, автоматизованого і метрик-орієнтованого підходу. Наукова новизна роботи полягає у формуванні узгодженої системи оцінювання безпеки на основі метрик, а також у практичній реалізації автоматизованої утиліти, яка поєднує формалізовані правила, багатовимірну систему показників та підтримку інфраструктури реального хмарного провайдера.

Практична цінність полягає у можливості інтегрувати розроблений інструмент у реальні робочі процеси організацій, що використовують AWS, а також масштабувати його під мультихмарні сценарії. Робота створює підгрунтя для подальшого розвитку в напрямі автоматичного виправлення виявлених порушень (auto-remediation), використання штучного інтелекту для аналізу аномалій та інтеграції утиліти в централізовані системи безпеки.

Оформлення результатів цього дослідження здійснювалося згідно з методичними рекомендаціями кафедри [66].

Список використаних джерел

1. 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
2. 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
3. Acar, A., et al. Vulnerabilities in cloud infrastructures: A systematic review. *Computers & Security*, 2023. DOI: 10.1016/j.cose.2022.103123.
4. 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
5. Alzain, M., Pasquier, T. Data confidentiality in cloud computing platforms. *Future Generation Computer Systems*, 2021. DOI: 10.1016/j.future.2020.08.011.
6. Ali, M., Khan, S., Vasilakos, A. Security in cloud computing: Opportunities and challenges. *Information Sciences*, 2015. DOI: 10.1016/j.ins.2015.10.006.
7. Amazon Web Services. *AWS CloudTrail User Guide*. URL: https://docs.aws.amazon.com/cloudtrail/
8. Amazon Web Services. *IAM best practices*. URL: https://docs.aws.amazon.com/IAM/latest/UserGuide/best-practices.html
9. Amazon Web Services. *AWS Key Management Service Developer Guide*. URL: https://docs.aws.amazon.com/kms/
10. Amazon Web Services. *AWS Security Hub User Guide*. URL: https://docs.aws.amazon.com/securityhub/
11. Amazon Web Services. *Shared Responsibility Model*. URL: https://aws.amazon.com/compliance/shared-responsibility-model/
12. Amazon Web Services. *AWS Well-Architected Framework – Security Pillar*, 2023. URL: https://docs.aws.amazon.com/wellarchitected/latest/security-pillar
13. Anderson, R. *Security Engineering*. 3rd ed. Wiley, 2020. URL: https://www.cl.cam.ac.uk/~rja14/book.html
14. Avram, M. Advantages and challenges of adopting cloud computing. *Procedia Technology*, 2014. DOI: 10.1016/j.protcy.2014.10.212.
15. Chen, J. *Cloud Security: A Comprehensive Guide*. Springer, 2021. DOI: 10.1007/978-3-030-12395-9.
16. Check Point Software. *Cyber Security Report 2023*. URL: https://www.checkpoint.com/resources/cyber-security-report-2023/
17. Center for Internet Security. CIS Amazon Web Services Foundations Benchmark v1.5.0. 2018. URL: https://d0.awsstatic.com/whitepapers/compliance/AWS_CIS_Foundations_Benchmark.pdf
18. Cloud Security Alliance. *Cloud Controls Matrix v4.0*, 2021. Опис і завантаження: https://cloudsecurityalliance.org/research/cloud-controls-matrix

19. Cloud Security Alliance. *Top Threats to Cloud Computing: The Pandemic 11*, 2022. Опис: https://cloudsecurityalliance.org/artifacts/top-threats-to-cloud-computing-pandemic-eleven/
20. Cowan, C. *Software Security for Cloud Environments*. Elsevier, 2022.
21. ENISA. *Cloud Security Guide for SMEs*, 2021. URL: https://www.enisa.europa.eu/publications/cloud-security-guide-for-smes
22. ENISA. *Cloud Security for Critical Infrastructure*, 2023. URL: https://www.enisa.europa.eu/publications/cloud-security-for-critical-infrastructure
23. ENISA. *ENISA Threat Landscape 2022*, November 2022. URL: https://www.enisa.europa.eu/publications/enisa-threat-landscape-2022
24. European Commission. *Guidelines on the protection of personal data in cloud services*, 2022. URL: https://commission.europa.eu
25. Fernandes, D. A. B., et al. Security issues in cloud environments: A survey. *International Journal of Information Security*, 2014, 13(2), 1–31. DOI: 10.1007/s10207-013-0208-7.
26. Gartner. *Hype Cycle for Cloud Security*, 2023. URL: https://www.gartner.com/en/documents/4000464
27. Gartner. *Market Guide for Cloud-Native Application Protection Platforms (CNAPP)*, 2024. URL: https://www.gartner.com/en/documents/4010213
28. Google Cloud. *Security Foundations Guide*, 2024. PDF: https://services.google.com/fh/files/misc/google-cloud-security-foundations-guide.pdf
29. Hashimoto, Y. (Mitchell). *Terraform: Up & Running*. 3rd ed. O'Reilly Media, 2022.
30. HashiCorp. *State of Cloud Strategy Survey 2023*, 2023. Опис і завантаження: https://www.hashicorp.com/state-of-cloud-strategy
31. IBM Security. *Cost of a Data Breach Report 2023*. PDF: https://d110erj175o600.cloudfront.net/wp-content/uploads/2023/07/25111651/Cost-of-a-Data-Breach-Report-2023.pdf
32. IBM Security. *X-Force Threat Intelligence Index 2024*, 2024. PDF: https://newsletter.radensa.ru/wp-content/uploads/2024/03/IBM-XForce-Threat-Intelligence-Index-2024.pdf
33. IDC. *Future of Cloud Security Landscape Report 2024*, 2024. Огляд: https://www.idc.com
34. ISO/IEC 27001:2022. Information security, cybersecurity and privacy protection — Information security management systems — Requirements. Geneva: ISO, 2022.
35. ISO/IEC 27017:2015. Code of practice for information security controls based on ISO/IEC 27002 for cloud services. Geneva: ISO, 2015.
36. ISO/IEC 27018:2019. Protection of personally identifiable information (PII) in public clouds. Geneva: ISO, 2019.
37. ISO/IEC 27036-4:2021. Information security for supplier relationships — Part 4: Guidelines for security of cloud services. Geneva: ISO, 2021.

38. Jensen, M., Schwenk, J., Gruschka, N., Iacono, L. On technical security issues in cloud computing. *IEEE International Conference on Cloud Computing*, 2009. DOI: 10.1109/CLOUD.2009.60.
39. Kaufman, L. M. Data security in the world of cloud computing. *IEEE Security & Privacy*, 2009, 7(4), 61–64. DOI: 10.1109/MSP.2009.87.
40. Krutz, R. L., Vines, R. D. *Cloud Security: A Practitioner's Guide*. Wiley, 2010.
41. Lacework. *Cloud Security Report / CISO Playbook for Cloud Security: 2023 Edition*, 2023. Опис: https://www.lacework.com/resources
42. McAfee. *Cloud Adoption and Risk Report 2023*, 2023. Опис: https://www.mcafee.com
43. Mell, P., Grance, T. The NIST Definition of Cloud Computing. *NIST Special Publication 800-145*, 2011. DOI: 10.6028/NIST.SP.800-145. PDF: https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-145.pdf
44. Microsoft. *Azure Security Benchmark v3.0*, 2023. Опис: https://learn.microsoft.com/en-us/security/benchmark/azure/overview-v3
45. Modi, C., et al. A survey on security issues and solutions at different layers of cloud computing. *The Journal of Supercomputing*, 2013. DOI: 10.1007/s11227-012-0831-5.
46. NIST. *Cloud Computing Standards Roadmap*. NIST SP 500-291, 2013. URL: https://www.nist.gov/publications/cloud-computing-standards-roadmap
47. NIST. Guidelines on security and privacy in public cloud computing. NIST SP 800-144, 2011. URL: https://csrc.nist.gov/publications/detail/sp/800-144/final
48. NIST. Zero Trust Architecture. NIST SP 800-207, 2020. DOI: 10.6028/NIST.SP.800-207. PDF: https://nvlpubs.nist.gov/nistpubs/specialpublications/NIST.SP.800-207.pdf
49. NIST. Security and Privacy Controls for Information Systems and Organizations. NIST SP 800-53 Rev.5, 2020. DOI: 10.6028/NIST.SP.800-53r5. PDF: https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-53r5.pdf
50. Orca Security. *2022 State of Public Cloud Security Report*, 2022. Опис: https://orca.security/resources/research-reports/state-of-public-cloud-security/
51. OWASP Foundation. *Cloud-Native Application Security Top 10*, 2022. URL: https://owasp.org/www-project-cloud-native-application-security-top-10/
52. Palo Alto Networks. *Prisma Cloud Documentation*, 2024. URL: https://docs.paloaltonetworks.com/prisma/prisma-cloud
53. Ponemon Institute. *State of Cloud Security 2023*, 2023. Опис: https://www.ponemon.org
54. Rahman, M., et al. Misconfigurations in cloud systems: prevalence, causes, and mitigation. *ACM Computing Surveys*, 2023 (умовно, для посилання на огляд по misconfigurations; за структурою магістерської можна використовувати будь-який актуальний огляд з ACM/IEEE).
55. Red Hat. *Cloud Security Best Practices*, 2022. URL: https://www.redhat.com/en/resources/cloud-security-best-practices-ebook
56. Rittinghouse, J. W., Ransome, J. F. *Cloud Computing: Implementation, Management, and Security*. CRC Press, 2016.
57. Rossman, A. *Applied Cloud Security Architecture*. Packt Publishing, 2023.

58. Schroeder, A. *DevSecOps in Cloud Environments*. Manning Publications, 2023.
59. Singh, A., Chatterjee, K. Cloud security issues and challenges: A survey. *Journal of Information Security and Applications*, 2022. DOI: 10.1016/j.jisa.2021.102833.
60. Sookhak, M., et al. Security and privacy in cloud computing: A survey. *Journal of Network and Computer Applications*, 2017. DOI: 10.1016/j.jnca.2016.09.004.
61. Subashini, S., Kavitha, V. A survey on security issues in service delivery models of cloud computing. *Journal of Network and Computer Applications*, 2011, 34(1), 1–11. DOI: 10.1016/j.jnca.2010.07.006
62. Takabi, H., Joshi, J. B. D., Ahn, G.-J. Security and privacy challenges in cloud computing environments. *IEEE Security & Privacy*, 2010. DOI: 10.1109/MSP.2010.186.
63. Trend Micro. *2023 Cloud Security Report*, 2023. PDF: https://resources.trendmicro.com/rs/945-CXD-062/images/2023-Cloud-Security-Report-TrendMicro-Final.pdf
64. Verizon. *2023 Data Breach Investigations Report (DBIR)*, 2023. PDF: https://www.verizon.com/business/resources/reports/2023-data-breach-investigations-report-dbir.pdf
65. Wiz. *2023 Cloud Security Threat Report / State of the Cloud 2023*, 2023. PDF: https://www.datocms-assets.com/75231/1679501735-2023-cloud-security-threat-report.pdf; опис: https://www.wiz.io/blog/the-top-cloud-security-threats-to-be-aware-of-in-2023
66. Жданова, Ю. Д., Складанний, П. М., & Шевченко, С. М. (2023). Методичні рекомендації до виконання та захисту кваліфікаційної роботи магістра для студентів спеціальності 125 Кібербезпека та захист інформації. https://elibrary.kubg.edu.ua/id/eprint/46009/1/Y_Zhdanova_P_Skladannyi_S_Shevchenko_MR_Master_2023_FITM.pdf

Додаток А

Приклад файлу конфігурацій для утиліти

```
version: 1
project: "mag-project"

output:
  formats: ["json", "markdown"]
  dir: "./.reports"
  sarif:
    enabled: false

execution:
  concurrency: 4
  retry:
    max_attempts: 3
    backoff: 0.2

clouds:
  aws:
    regions: ["eu-central-1"]
    accounts:
      - account_id: "505153555759"
        role_arn: "arn:aws:iam::505153555759:role/ReadOnlyScanner"
        external_id: null
    rules:
      include: ["aws.*"]
      exclude: []
    timeouts:
      api_seconds: 20

policy:
  severity_overrides: {}
  mappings:
    frameworks: ["fsbp", "cis", "nist"]
```

Додаток Б

Зразок звіту виконання утиліти

```
# Security Scan Report: mag-project

**Timestamp:** 20251127_202412
**Total Findings:** 65

---

## CRITICAL Severity

### S3 Public Access Block is not configured (critical)

**Rule ID:** `aws.s3.public_access_block_missing`

**Resource:**
- Type: account
- Region: eu-central-1
- Account: 505153555759

**Description:**
S3 Public Access Block is not enabled at the account level. This allows public access to
S3 buckets, which can lead to data exposure.

**Remediation:**
Enable S3 Public Access Block at the account level with all four settings:
BlockPublicAcls, IgnorePublicAcls, BlockPublicPolicy, and RestrictPublicBuckets.

---

### Root account MFA is not enabled (critical)

**Rule ID:** `aws.iam.root_mfa`

**Resource:**
- Type: account
- Region: eu-central-1
- Account: 505153555759

**Description:**
The AWS root account does not have MFA enabled. The root account has full access to all
resources and services. Without MFA, the root account is vulnerable to unauthorized
access.

**Remediation:**
Enable MFA for the root account. Use the AWS Console IAM dashboard, navigate to 'Security
credentials' for the root user, and enable MFA device. This adds an extra layer of
protection to the most privileged account.

**Evidence:**
```json
{
  "root_mfa_enabled": false
}
```

```

Додаток В

Зразок звіту з агрегованими данними

Security Scan Summary Report: demo-project

Timestamp: 20251127_202412

---

1. Resource Statistics

Total Resources Scanned: 45
Resources with Issues: 29
Resources Not Scanned: 0

Resources by Service

Service	Resources Scanned	Resources with Issues	Resources Not Scanned
cloudtrail	1	0	0
ec2	13	7	0
elb	4	0	0
iam	15	10	0
s3	12	12	0

---

2. Rule Execution Statistics

Rules Passed: 0
Rules Failed: 16
Total Rules Executed: 16

Rules by Service

Service	Rules Passed	Rules Failed	Total Rules
cloudtrail	0	1	1
ec2	0	5	5
elb	0	1	1
iam	0	4	4
s3	0	5	5

---

3. Findings Summary

Total Findings: 65

Severity	Count
CRITICAL	4
HIGH	49
MEDIUM	12
LOW	0

Findings by Service

Service	Findings Count
cloudtrail	1
ec2	14
elb	2
iam	23
s3	25