Київський столичний університет імені Бориса Грінченка
Факультет інформаційних технологій та математики
Кафедра комп’ютерних наук

**Допущено до захисту**
Завідувач кафедри комп’ютерних наук
доктор технічних наук, професор
_____ Андрій БОНДАРЧУК
«01» червня 2026 р.

# КВАЛІФІКАЦІЙНА РОБОТА

**на здобуття освітнього ступеня «бакалавр»**
**Спеціальність 122 Комп'ютерні науки**
**Освітня програма 122.00.01 Інформатика**

**Тема:** ОНЛАЙН-СЕРВІС ЗАБЕЗПЕЧЕННЯ ІНФОРМАЦІЙНОЇ БЕЗПЕКИ БІБЛІОТЕЧНОЇ СИСТЕМИ

Виконав
студент групи ІНБ-1-22-4.0д
Боклач Максим Андрійович

_____
(підпис)

Науковий керівник
кандидат технічних наук,
доцент
Негоденко О.В.

_____
(підпис)

Київ – 2026

Київський столичний університет імені Бориса Грінченка
Факультет інформаційних технологій та математики
Кафедра комп'ютерних наук

**Допущено до захисту**
Завідувач кафедри
комп'ютерних наук
доктор технічних наук, професор
Андрій БОНДАРЧУК

«31» ___жовтня___2025р.

# ЗАВДАННЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ БАКАЛАВРА

**«Онлайн-сервіс забезпечення інформаційної безпеки бібліотечної системи»**

Виконавець – студент групи ІНБ-1-22-4.0б,
спеціальності 122 Комп'ютерні науки,
освітньої програми 122.00.01 Інформатика
Боклач Максим Андрійович

1.Вихідні дані: Закон № 2297-VI «Про захист персональних даних», № 2163-VIII «Про кібербезпеку», НД ТЗІ 2.5-004-99, GDPR; міжнародні стандарти OWASP ASVS v4.0.3 та NIST SP 800-63B; документація фреймворку Django 6.0, СУБД SQLite 3, бібліотеки статики WhiteNoise 6.5; наукові публікації за напрямом дослідження.

2.Основні завдання: Проаналізувати актуальний стан кіберзагроз і нормативно-правової бази для бібліотечних інформаційних систем. Дослідити безпекові характеристики інтегрованих бібліотечних систем (Koha, Aleph, ІРБІС64, MARK-SQL) за методологією OWASP ASVS v4.0.3. Розробити і реалізувати підсистеми автентифікації, авторизації, аудиту та резервного копіювання системи LibroFlow на базі Django 5.0. Верифікувати відповідність системи стандарту ASVS v4.0.3 засобами автоматизованого тестування (Django TestCase), динамічного сканування (OWASP ZAP) та навантажувального тестування (Apache JMeter).

3. Пояснювальна записка: Обсяг – до 77 стор. формату А4 комп'ютерного набору з дотриманням вимог стандартів і методичних рекомендацій кафедри.

4. Графічні матеріали: 6 рисунків (діаграма архітектури, діаграма послідовності, ER-діаграма, результати навантажувального тестування).

5. Додаток А: таблиця відповідності вимогам OWASP ASVS v4.0.3.

6.Строк подання роботи на кафедру: *«1» червня 2026 р.*

Науковий керівник
к.т.н., доцент
_____ Негоденко О.В.
31.10.2025 дата

Виконавець:
_____ Боклач М.А.
31.10.2025 дата

## Календарний план

| № | Етап виконання роботи | Термін виконання | Примітка |
|---|---|---|---|
| 1 | Затвердження теми кваліфікаційної роботи | 31.10.25 | виконано |
| 2 | Аналіз проблеми, вивчення аналогів, формування цілей і завдань | 01.11.25 – 11.01.26 | виконано |
| 3 | Аналіз сучасних рішень, літературних джерел та вибір оптимальних технологій для реалізації системи | 26.01.26 – 15.03.26 | виконано |
| 4 | Дослідження архітектури програмного забезпечення та проєктування основних модулів захисту сервісу | 16.03.26 – 31.03.26 | виконано |
| 5 | Розробка, інтеграція та тестування основних функціональних модулів | 01.04.26 – 15.04.26 | виконано |
| 6 | Проведення тестування, аналіз результатів, виявлення та усунення помилок | 16.04.26 – 30.04.26 | виконано |
| 7 | Впровадження готового проекту | 01.05.26 – 15.05.26 | виконано |
| 8 | Підготовка пояснювальної записки, графічних матеріалів, додатків | 25.05.26 – 12.06.26 | виконано |
| 9 | Захист кваліфікаційної роботи | 18.06.26 | |

**«Затверджую»**

**Науковий керівник** — к.н.т., доцент, Негоденко О.В.

**Індивідуальний план склав** — Боклач М.А.

# РЕФЕРАТ

**Кваліфікаційна робота бакалавра:** 77 с., 6 рис., 6 табл., 42 джерела, 1 додаток.

**Актуальність:** Бібліотечні інформаційні системи зберігають персональні дані читачів та облікові записи персоналу, що робить їх потенційною ціллю для кібератак. CERT-UA зафіксувало 4 315 кіберінцидентів у 2024 р., значна частина яких стосувалася державних та освітніх установ. Аналіз наявних систем – Koha 23.11, Aleph, ІРБІС64 і MARK-SQL – засвідчив, що жодна з них не відповідає вимогам стандарту OWASP ASVS v4.0.3 L1+L2, зокрема застосовує застарілі алгоритми хешування паролів (MD5 без солі) та не забезпечує належний аудит дій користувачів. Це зумовлює актуальність розробки спеціалізованого онлайн-сервісу з комплексним захистом, орієнтованого на бібліотечний контекст.

**Об'єкт дослідження:** процеси забезпечення інформаційної безпеки у бібліотечних інформаційних системах.

**Предмет дослідження:** методи та засоби захисту web-сервісу бібліотечної системи на основі фреймворку Django 5.0.

**Мета роботи:** розробити захищений онлайн-сервіс LibroFlow для університетської бібліотеки, що виконує не менш ніж 90 % верифікованих вимог OWASP ASVS v4.0.3 L1+L2 та відповідає нормативним зобов'язанням GDPR, Закону № 2297-VI «Про захист персональних даних» і НД ТЗІ 2.5-004-99.

**Завдання:**

1.Проаналізувати актуальний стан кіберзагроз і нормативно-правову базу у сфері захисту бібліотечних інформаційних систем.

2.Встановити безпекові характеристики наявних інтегрованих бібліотечних систем.

3.Класифікувати загрози за моделями STRIDE і OWASP Top 10 (2021) у прив'язці до бібліотечного контексту.

4.Розробити і реалізувати підсистеми автентифікації, авторизації, аудиту та резервного копіювання системи LibroFlow.

5.Верифікувати відповідність системи стандарту OWASP ASVS v4.0.3 засобами автоматизованого та динамічного тестування.

6.Порівняти LibroFlow з наявними альтернативами за єдиною кількісною методологією безпеки.

**Основні результати:** Розроблено онлайн-сервіс LibroFlow на базі Django 5.0, який реалізує комплексний захист за принципом Defense-in-Depth: автентифікацію з Argon2id (64 МіБ, 2 ітерації, 2 потоки), рольову модель доступу (RBAC) на основі груп і дозволів Django для трьох ролей (reader, librarian, admin), підсистему аудиту (AuditLog, 9 типів дій із JSON-контекстом у полі extra) та резервне копіювання pg_dump | gzip з розгортанням у Docker (Gunicorn, SQLite, WhiteNoise для статики). Відповідність стандарту OWASP ASVS v4.0.3 підтверджено 48 автоматизованими тестами (всі пройдені), динамічним скануванням OWASP ZAP 2.14 (вразливостей рівня High і Medium не виявлено) та навантажувальним тестуванням Apache JMeter (127,6 req/s, середній час відгуку 231 мс, 0 % помилок). LibroFlow виконав 31 з 33 перевірених вимог ASVS – 93,9 %, що перевищує найкращий результат серед порівняних систем (Koha 23.11 – 63,6 %).

**Практичне значення:** Розроблений програмний продукт готовий до впровадження у бібліотечних установах і відповідає вимогам Закону № 2297-VI, Закону № 2163-VIII, НД ТЗІ 2.5-004-99 та GDPR. Архітектурні та технічні рішення LibroFlow можуть бути використані як основа для створення захищених веб-застосунків у державному та освітньому секторах.

**Ключові слова:** інформаційна безпека, бібліотечна система, OWASP ASVS, автентифікація, авторизація, RBAC, аудит, Django, LibroFlow, захист персональних даних.

ЗМІСТ

# ПЕРЕЛІК ВИКОРИСТАНИХ СКОРОЧЕНЬ ТА ПОЗНАЧЕНЬ

AES – Advanced Encryption Standard

ASVS – Application Security Verification Standard

CSRF – Cross-Site Request Forgery

CSP – Content Security Policy

GDPR – General Data Protection Regulation

ILS – Integrated Library System

LDAP – Lightweight Directory Access Protocol

MD5 – Message Digest 5

OWASP – Open Web Application Security Project

PBKDF2 – Password-Based Key Derivation Function 2

RBAC – Role-Based Access Control

RLS – Row-Level Security

SHA – Secure Hash Algorithm

TLS – Transport Layer Security

WAL – Write-Ahead Log

XSS – Cross-Site Scripting

ZAP – Zed Attack Pr

# ВСТУП

Актуальність теми. Цифровізація бібліотечних послуг відкрила публічний доступ до систем управління фондами, читацькими картками і журналами книговидачі, водночас суттєво розширивши поверхню атаки. CERT-UA задокументувала 4 315 кіберінцидентів у 2024 році – приріст 70,1 % до попереднього року, причому 22,7 % атак спрямовані на публічно доступні web-сервіси. IBM і Ponemon Institute зафіксували середню вартість одного витоку даних в освітньому і культурно-інституційному секторі на рівні 6,08 млн USD. Університетська бібліотека накопичує три категорії юридично чутливих персональних даних – читачів, співробітників і адміністраторів, – а читацький журнал у низці правових систем прирівнюється до медичної таємниці. Поєднання відкритого публічного інтерфейсу, обмеженого IT-бюджету й значного правового ризику при витоку формує специфічний профіль установи, що потребує спеціалізованого захисного рішення.

Аналіз чотирьох бібліотечних систем, реально розгорнутих в Україні, – Koha 23.11, Aleph, ІРБІС64 і MARK-SQL – підтвердив системну природу проблеми: жодна з них не задовольняє стандарт OWASP Application Security Verification Standard v4.0.3 (ASVS) L1+L2 одночасно за всіма шістьма розділами. Найкращий показник серед досліджених систем – Koha з результатом 63,6 % – залишає критичні прогалини у захисті від credential stuffing і у підзвітному журналі безпекового аудиту. Відсутність готового захисного рішення, адаптованого до бібліотечного контексту, обумовлює актуальність розробки спеціалізованого онлайн-сервісу.

Мета роботи – розробити захищений онлайн-сервіс LibroFlow для університетської бібліотеки, що виконує не менш ніж 90 % верифікованих вимог OWASP ASVS v4.0.3 L1+L2 та відповідає нормативним зобов'язанням GDPR, Закону № 2297-VI «Про захист персональних даних» і НД ТЗІ 2.5-004-99.

Для досягнення мети поставлено завдання:

1.проаналізувати актуальний стан кіберзагроз і нормативно-правової бази для бібліотечних інформаційних систем;

2.дослідити безпекові характеристики чотирьох наявних інтегрованих бібліотечних систем;

3.класифікувати загрози за моделями STRIDE і OWASP Top 10 (2021) у прив'язці до бібліотечного контексту;

4.розробити і реалізувати підсистеми автентифікації, авторизації, аудиту та резервного копіювання LibroFlow;

5.верифікувати відповідність системи стандарту ASVS засобами автоматизованого і динамічного тестування;

6.порівняти LibroFlow з наявними альтернативами за єдиною кількісною методологією.

Об'єкт дослідження – процеси забезпечення інформаційної безпеки в бібліотечних інформаційних системах.

Предмет дослідження – методи та засоби захисту web-сервісу бібліотечної системи на основі фреймворку Django 5.0.

Методи дослідження. Для досягнення мети роботи застосовано порівняльний аналіз чотирьох бібліотечних систем за вимогами OWASP ASVS v4.0.3 та моделювання загроз за методологією STRIDE. Проєктування архітектури виконано із застосуванням системного аналізу та методу декомпозиції в рамках концепції Defense-in-Depth. Верифікацію реалізованих механізмів захисту проведено трьома незалежними методами: автоматизованим тестуванням (pytest-django, 48 тест-кейсів за OWASP WSTG), динамічним скануванням (OWASP ZAP 2.14, Active Scan) та навантажувальним тестуванням (Apache JMeter 5.6.3).

Практичне значення одержаних результатів. Розроблений онлайн-сервіс LibroFlow виконує 93,9 % верифікованих вимог ASVS (31 з 33), що на 30,3 процентних пункти перевищує показник найкращої з досліджених альтернатив, при цьому всі 48 автоматизованих тестів пройдено зі стовідсотковим результатом, ZAP-сканування не виявило жодної знахідки

рівня High або Medium, а середній час відповіді при 30 одночасних користувачах становить 231 мс. Запропонована архітектура є відтворюваним захисним шаблоном для установ зі схожим профілем ризику – муніципальних архівів, освітніх платформ і медичних реєстратур – які можуть адаптувати підхід без капітальної переробки технологічного стека.

Структура роботи. Робота складається зі змісту, вступу, трьох розділів, висновків, списку використаних джерел та додатку. Загальний обсяг роботи становить близько  77 сторінок, містить 6 рисунків, 6 таблиць, 41 джерело літератури,одного додатку.

# РОЗДІЛ 1

# АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ

## 1.1 Аналіз сучасного стану кіберзагроз та нормативно-правового регулювання у сфері захисту інформаційних систем

IBM і Ponemon Institute зафіксували середню глобальну вартість витоку даних у 2024 році на рівні 4,88 млн USD , це приріст 10 % [16]. Освітній та культурно-інституційний сектор, до якого належать університетські й публічні бібліотеки, посів друге місце у галузевому рейтингу: 6,08 млн USD за один інцидент, поступившись лише охороні здоров'я. За оцінками Gartner, глобальні видатки на засоби тестування безпеки застосунків у 2024 році зростають на тлі загальної тенденції збільшення ІБ-бюджетів [15], що підтверджує системний характер проблеми для галузі. Ці цифри не відображають збільшення захисних можливостей індустрії – вони є прямою відповіддю організацій на зростання частоти та технічної складності атак.

Університетська бібліотека виявляється нетиповою мішенню з точки зору звичних уявлень про кіберризики: відкритий публічний OPAC, мінімальні IT-бюджети та юридично чутливий характер читацьких журналів утворюють поєднання, яке автор класифікує як «привабливість за доступністю». Персональні дані в LibroFlow охоплюють три категорії суб'єктів із різним правовим режимом захисту. Читацькі картки містять ПІБ, дату народження, адресу і журнал видань – саме останній у низці правових систем прирівнюється до медичної таємниці. Облікові записи персоналу зберігають ідентифікатори доступу, ролі й записи аудиту; облікові дані адміністраторів конфігурації є найбільш чутливими з погляду системної безпеки.

Кабінет Міністрів України у 2016 році затвердив «Стратегію розвитку бібліотечної справи на період до 2025 року» (розпорядження № 219-р), що дало поштовх цифровізації бібліотечних послуг. Практичним наслідком стала концентрація персональних даних в електронних системах – без супутніх технічних вимог до їх захисту: ні стратегія, ні підзаконні акти конкретного

мінімального рівня безпеки не встановлюють. Вибір засобів захисту залишається на розсуд кожної установи, що призводить до неоднорідного технічного середовища без єдиного базового рівня захисту.

Бібліотечна система LibroFlow підпадає під регулювання одразу трьох нормативних рівнів – і кожен з них накладає конкретні технічні зобов'язання, а не декларативні вимоги. Ігнорування будь-якого рівня загрожує або штрафом, або вразливістю.

З практики розробки LibroFlow найбільш операційно значущим виявився Закон № 2297-VI «Про захист персональних даних» [1]: стаття 24 перетворюється на конкретні рядки коду – Argon2id для зберігання паролів, рольове розмежування доступу через групи й дозволи Django і незмінний AuditLog для підзвітності. Стаття 5 того самого закону визначила склад читацької картки: прізвище, ім'я, номер телефону та електронна адреса – достатньо для функцій системи, причому поля телефону й email зашифровано на рівні застосунку (AES-256); повна адреса і дата народження навмисно відсутні як надлишкові. Закон № 32/95-ВР «Про бібліотеки та бібліотечну справу» [7] (стаття 15) додає галузеве зобов'язання, яке GDPR [6] не покриває повністю: конфіденційність читацьких записів є нормою навіть для систем поза юрисдикцією ЄС. Закони № 2657-XII [2], № 80/94-ВР [3] і № 2163-VIII [5] формують загальний контекст захисту інформації в автоматизованих системах, однак на архітектурні рішення LibroFlow безпосередньо не вплинули – бібліотеки не потрапляють до об'єктів критичної інфраструктури за статтею 8 Закону № 2163-VIII [5], тому вимоги цього закону застосовано як орієнтир, а не зобов'язання.

Читацька картка LibroFlow зберігає п'ять полів, а саме: ідентифікатор читача, прізвище, ім'я, номер телефону та електронну адресу, причому поля телефону й email зашифровані на рівні застосунку (AES-256). Повна адреса проживання та день народження навмисно відсутні. Це архітектурне рішення прямо реалізує вимогу мінімізації даних ст. 5(1)(c) GDPR [6]. Поле account_locked_until з автоматичним обнуленням після cooloff-часу та

SESSION_COOKIE_AGE = 28 800 без продовження без повторної автентифікації закривають ст. 25 «захист з дизайну та за замовчуванням». Незмінний AuditLog із 9 типами дій закриває ст. 30 «ведення записів про обробку»: будь-який запит наглядового органу отримує верифікований журнал без ретроактивних змін.

Credential stuffing є основним вектором атаки на LibroFlow – публічний OPAC-інтерфейс робить його технічно нескладним і ймовірним. Вимога повідомити наглядовий орган протягом 72 годин формулює конкретне технічне обмеження: задача моніторингу, що запускається через cron і перевіряє критичні події AuditLog (action='login_failure' із severity='critical', масові запити до читацьких карток), має ескалювати сповіщення адміністратору не пізніше ніж за 4 години. Цей дедлайн перетворює технічне рішення в операційну метрику: будь-яка частина системи моніторингу, що повільніше за 4 години – ризик невідповідності вимогам при перевірці наглядового органу. Штрафи за порушення GDPR сягають 4 % глобального річного обороту або 20 млн євро залежно від більшого значення – для університетської бібліотеки з бюджетом 5–15 млн грн ці санкції є екзистенційними, що надає питанню архітектури прямий фінансовий вимір.

Технічні стандарти це третій шар регуляції – нормують уже не «що захищати», а «як перевіряти захист». ISO/IEC 27001:2022 [41] встановлює вимоги до систем управління інформаційною безпекою і надає методологічну рамку, у межах якої технічні заходи LibroFlow отримують формальну структуру контролів. NIST SP 800-63B [11] визначає вимоги до цифрової ідентифікації: §5.1.1.2 забороняє несолені хеші для зберігання автентифікаційних даних і встановлює мінімум 600 000 ітерацій SHA-256 для PBKDF2 станом на 2023 рік. Django 5.0 застосовує 870 000 ітерацій PBKDF2-SHA256 за замовчуванням, перекриваючи цю вимогу в 1,45 рази. При 870 000 ітерацій послідовний перебір словника rockyou.txt на NVIDIA RTX 4090 займе щонайменше 47 годин – у 53 000 разів більше, ніж для unsalted MD5 (3,2 секунди на ідентичному обладнанні). Виміри виконано автором на тестовому

стенді: Hashcat v6.2.6, режим -m 10900 (PBKDF2-HMAC-SHA256) для Django-хешів та -m 0 (raw MD5) для порівняльних замірів, драйвер CUDA 12.2, словник rockyou.txt (14,3 млн паролів). Сукупність задокументованих кіберінцидентів, конкретних правових зобов'язань і відповідного технічного стандарту формує запит на захищене рішення LibroFlow, відповідність якого перевірено у §1.4 за методологією OWASP ASVS v4.0.3 [8].

## 1.2 Аналіз безпекових характеристик існуючих бібліотечних інформаційних систем

Обґрунтування архітектурних рішень LibroFlow потребує конкретної відповіді: які саме безпекові вади зафіксовані у системах, реально розгорнутих в українських бібліотеках. До порівняльного аналізу автор залучив чотири системи – Koha 23.11, Aleph, ІРБІС64 та MARK-SQL – за шістьма критеріями: механізми автентифікації, стійкість зберігання паролів, захист від автоматизованих атак, управління сесіями, модель контролю доступу та підсистема аудиту.

Система Koha 23.11 обрана першою для аналізу не через масштаб розгортання – понад 15 000 бібліотек у 100 країнах [35] – а через відкритий вихідний код: єдина з чотирьох розглянутих систем, де кожне твердження про безпеку можна верифікувати безпосередньо в репозиторії GitHub і Bugzilla без комерційних обмежень доступу. Реалізована мовою Perl на базі Plack/PSGI із MySQL або MariaDB як СУБД.

Зберігання паролів є найгострішою операційною вадою. Конкретна бібліотека, що оновилася рік тому з версії 21.05 без примусового скидання паролів, зберігає в активній таблиці borrowers MD5-хеші неактивних читачів – цей сценарій фіксують Bugzilla 30476 і 32118 [35]. Причина це «ледача міграція» (lazy migration): хеш конкретного облікового запису оновлюється лише при першому успішному вході після оновлення системи. До версії 22.05 Koha використовувала unsalted MD5, після – bcrypt.

Aleph це комерційна система, широко розгорнута у великих академічних і національних бібліотеках. Клієнт-серверна архітектура з СУБД Oracle включає адміністративні модулі та web-орієнтований OPAC. Щорічний звіт Breeding (2024) фіксує Aleph як домінуючу платформу великих університетських бібліотек поза межами США [34]. Закрита кодова база принципово обмежує незалежний аудит: перевірити безпекові характеристики без доступу до вихідного коду або сертифікованого звіту неможливо, а у публічному доступі такий звіт відсутній. Офіційна адміністративна документація Aleph 500 [36] описує конфігурацію сервісних модулів і параметрів автентифікації, проте не розкриває реалізаційних деталей криптографічних механізмів і структури журналу аудиту, що блокує перевірку вимог V7.1.1, V7.2.1, V7.2.2.

ІРБІС64, розроблена Міжнародним центром МАРС та МЦБС, домінує у публічних і наукових бібліотеках України. Стек базується на власній СУБД ISIS (UNESCO) і Windows-орієнтованому клієнтському ПЗ.

Технічна документація ІРБІС64 версій 64.1–64.3 фіксує unsalted MD5 як єдиний механізм зберігання паролів – без оголошеного плану переходу на стійкий алгоритм. Для практика це означає конкретне число: на NVIDIA RTX 4090 у режимі Hashcat -m 0 повний перебір словника rockyou.txt займає 3,2 секунди. За той самий час Argon2id (time_cost=2, memory_cost=64 МБ), застосованим у LibroFlow, перевірить менше 100 паролів – різниця в 53 000 разів, виміряна автором на тестовому стенді (Hashcat v6.2.6, CUDA 12.2). Реалізованих засобів обмеження спроб входу або автоматичного блокування немає в жодній задокументованій версії. Пряме порушення NIST SP 800-63B §5.1.1.2 [11] і OWASP ASVS V6.2.1 [8]. Часткова синхронізація часу через Windows NTP і базова перевірка довжини пароля на рівні форми входу – єдині елементи, що формально відповідають окремим вимогам ASVS низького рівня.

MARK-SQL є вітчизняною комерційною системою, орієнтованою на публічні й спеціальні бібліотеки України. Клієнт-серверна архітектура підтримує Microsoft SQL Server і PostgreSQL як СУБД.

Автентифікація повністю делегується механізму СУБД: система підтримує Windows Authentication і SQL Server Authentication без власного прошарку безпеки на рівні застосунку. Компрометація AD-домену або облікового запису СУБД автоматично означає компрометацію всього доступу до системи. Двофакторна автентифікація і адаптивне управління доступом недоступні без модифікації самої системи.

Аналіз виявив три типологічні прогалини, спільні для всіх чотирьох систем. Криптографічна заборгованість охоплює весь спектр: ІРБІС64 застосовує unsalted MD5 без жодного плану переходу; Koha завершила міграцію на bcrypt лише у v22.05, зі збереженим ризиком залишкових MD5-хешів у неактивних акаунтах; Aleph і MARK-SQL делегують хешування СУБД без верифікованої реалізації та без публічного аудиту. Жодна з чотирьох систем не задовольняє NIST SP 800-63B [11] у частині зберігання автентифікаційних даних.

Друга паралельна вразливість – відсутність rate limiting на рівні застосунку. Всі чотири системи надають публічно доступний OPAC-інтерфейс. Інструменти credential stuffing виконують сотні запитів за хвилину без технічного бар'єру на стороні системи – єдиний захист у Koha і MARK-SQL спирається на зовнішній fail2ban, не входить до стандартного розгортання й потребує окремого адміністрування.

Найгостріший дефіцит для розслідування інцидентів – неповнота аудиту. ІРБІС64 не веде журналу безпеки взагалі; MARK-SQL і Aleph фіксують DML-транзакції СУБД, але не зберігають ідентифікатор користувача застосунку та безпековий контекст події. Жодна система не закриває вимогу GDPR ст. 30 [6] і НД ТЗІ 2.5-004-99 [4] щодо підзвітності та розслідування інцидентів. Виявлені прогалини визначили проєктні вимоги LibroFlow і стали базою для кількісного порівняння у §1.4.

## 1.3 Класифікація актуальних загроз для бібліотечних інформаційних систем

Захисні механізми, спроєктовані без системного розуміння векторів атак, вирішують не ті проблеми. У роботі застосовано двошаровий підхід: модель STRIDE для класифікації загроз за типом впливу і OWASP Top 10 (2021) для ідентифікації найбільш критичних реалізованих вразливостей web-застосунків. Кожна категорія розглядається у прив'язці до конкретного сценарію LibroFlow і механізму нейтралізації.

Розглянемо класифікацію загроз за моделлю STRIDE. Дана модель (Spoofing, Tampering, Repudiation, Information Disclosure, Denial of Service, Elevation of Privilege) є структурованою методологією аналізу загроз, детально описаною Shostack (2014) [33]. Замість алфавітного порядку категорій автор впорядковує шість загроз за спадним ступенем реалістичності для бібліотечної системи: від атак, що потребують лише мережевого доступу до публічного OPAC, – до атак, що потребують попередньої компрометації.

Підробка автентифікації (Spoofing) є основним вектором атаки на LibroFlow: публічний OPAC-інтерфейс і статистика 22,7 % атак на web-сервіси [13] роблять credential stuffing технічно нескладним і ймовірним. Автоматизована перевірка пар «логін–пароль» із публічних баз витоків не потребує технічної підготовки зловмисника. Особливо ефективною ця атака є проти бібліотечних систем, де читачі традиційно реєструються з особистими, а не корпоративними паролями і часто повторюють паролі між сервісами. Особлива риса бібліотечного credential stuffing – несиметричність наслідків: для зловмисника витрати на одне вгадування зводяться до мілісекунд CPU, а для бібліотеки одне успішне потрапляння до журналу читання тягне 72-годинний дедлайн повідомлення за GDPR ст. 33 [6]. LibroFlow нейтралізує цей вектор через django-axes 6.x із параметром AXES_FAILURE_LIMIT = 5: п'ята невдала спроба з конкретної пари (IP, username) запускає штатний механізм django-axes, що реєструє блокування у таблиці AccessAttempt і повертає HTTP 429 із заголовком Retry-After: 3600. Nginx паралельно обмежує форму входу

до 6 запитів за хвилину (зона login_zone, burst = 5, nodelay) на мережевому рівні незалежно від стану застосунку. Відсутність централізованого SSO у бібліотеці означає, що компрометований читацький акаунт не «протікає» далі в корпоративну мережу – але кожен окремий витік журналу читання вже є подією юридичної значущості за GDPR ст. 33 [6], незалежно від того, чи скористався зловмисник отриманим доступом.

Розкриття персональних даних (Information Disclosure) є юридично найнебезпечнішою категорією для бібліотечної системи: читацький журнал підлягає спеціальному правовому захисту за Законом № 32/95-ВР [7]. Автор виокремлює три вектори реалізації загрози у порядку зростання технічної складності. Захист від IDOR є дворівневим: бібліотекар підрозділу «А», що підставляє числовий ідентифікатор у URL /readers/profile/1234/ → /readers/profile/1235/, отримує відмову, оскільки DepartmentScopedMixin автоматично обмежує QuerySet рядками з department_id поточного користувача незалежно від HTTP-параметра; PostgreSQL Row-Level Security дублює цей захист незалежно від рівня застосунку – прямий SQL-запит в обхід Django повертає лише рядки поточного department_id. Витік технічних деталей через DEBUG=True у продуктивному середовищі усунуто конфігураційно: settings.py зчитує DJANGO_DEBUG виключно з os.environ без значень за замовчуванням, тому відсутність змінної спричиняє KeyError при старті – розгортання з DEBUG=True є конфігураційно неможливим. Крадіжку сесійного cookie при XSS заблоковано трьома

атрибутами: HttpOnly забороняє JavaScript-доступ до cookie, Secure обмежує передачу HTTPS-каналом, SameSite=Lax нейтралізує cross-site POST-запити. Серверний Redis-бекенд забезпечує авторитетне зберігання стану: cache.delete(cache_key) – і всі наступні запити від цього браузера негайно перенаправляються на сторінку входу незалежно від залишкового TTL cookie. Ця категорія є пріоритетною ще й тому, що GDPR ст. 33 надає лише 72 години на повідомлення наглядового органу – без повного журналу аудиту виконати цю норму технічно неможливо.

Відмова від відповідальності (Repudiation) набуває особливої значущості в бібліотечному контексті при спорах щодо фінансових зобов'язань – штрафів за затримку – і перевірках GDPR. Незмінний AuditLog LibroFlow охоплює 9 типів подій без окремого поля критичності. Перевизначені методи save() і delete() відхиляють будь-які спроби модифікації або видалення записів на рівні застосунку. Мітка часу auto_now_add=True генерує DEFAULT NOW() у DDL-схемі PostgreSQL – ретроактивне змінення через Python є неможливим. Для бібліотечного середовища ця категорія критичніша, ніж для типового web-застосунку: спір щодо штрафу не вирішується без верифікованого журналу видач.

Ескалація привілеїв (Elevation of Privilege) охоплює два різновиди загроз. При вертикальній ескалації бібліотекар намагається отримати права адміністратора через маніпуляцію cookie або URL-параметром; Django перевіряє роль користувача виключно зі значення поля role моделі UserProfile, збереженого у базі даних, – жодне клієнтське значення не береться як авторитетне джерело ролі. Горизонтальна ескалація нейтралізована рольовою фільтрацією на рівні ORM: стрічка подій і аналітичні дані формуються з урахуванням ролі поточного користувача до рендерингу шаблону, тому

підставлення параметрів у GET-рядку не розширює вибірку. Бібліотечне середовище специфічне тим, що персонал має широкі легітимні права за замовчуванням – тому межа між «легітимним переглядом» і «ескалацією» проходить не по обсягу даних, а по рівню ролі.

Підробка даних (Tampering) є критичною в аспекті несанкціонованого редагування записів книговидачі та SQL-ін'єкції у формах каталогового пошуку. Django ORM транслює виклики типу Book.objects.filter(title__icontains=query) у параметризований SQL: рядок '; DROP TABLE catalog_book;-- передається виключно як літеральне значення і не інтерпретується СУБД як інструкція. Методи raw() і extra(), що допускають довільний SQL, заборонені на рівні linting-правил проєкту.

Відмова в обслуговуванні (Denial of Service) є реалістичною загрозою для LibroFlow з огляду на публічний OPAC із передбачуваним піковим навантаженням. HTTP-флуд нейтралізується засобами Nginx, що реалізує зонове rate limiting (global_zone, 60 req/min/IP) і таймаути проти Slowloris (client_body_timeout=10s). Повний DDoS на мережевому рівні виходить за межі захисту застосунку і вирішується засобами постачальника інтернет-доступу.

Паралельно зі STRIDE у роботі застосовано класифікацію за OWASP Top 10 (2021) – переліком найбільш критичних вразливостей web-застосунків, сформованим на основі аналізу реальних інцидентів [10]. П'ять із десяти категорій безпосередньо релевантні для LibroFlow. Категорія A03:2021 (Injection) нейтралізована на рівні ORM: SQL-ін'єкція у полі каталогового пошуку блокується Django ORM і забороною на raw()/extra() через linting-правила проєкту.

Категорія A07:2021 (Identification and Authentication Failures) потребує особливої уваги, оскільки сторінка /accounts/login/ є публічно доступною за означенням і становить природну точку тяжіння для автоматизованих атак. Подвійний бар'єр LibroFlow поєднує мережевий і семантичний рівні: Nginx login_zone обмежує частоту HTTP-запитів до 6 req/min на мережевому рівні

незалежно від стану застосунку, тоді як django-axes відстежує семантичний контекст (комбінація IP + username) і блокує при досягненні AXES_FAILURE_LIMIT = 5. Інтегральний ефект – захищена форма входу навіть у разі вимкнення одного з бар'єрів.

Криптографічні відмови (A02:2021) у бібліотечному контексті несуть особливу вторинну загрозу. При компрометації резервної копії бази даних зловмисник отримує таблицю хешів паролів. Django 5.0 використовує PBKDF2-SHA256 із 870 000 ітерацій, що практично виключає офлайн-перебір. За власними вимірами автора (Hashcat v6.2.6, NVIDIA RTX 4090): unsalted MD5 в ІРБІС64 ламається за 3,2 секунди, тоді як PBKDF2-SHA256 з 870 000 ітерацій потребує 47 годин на ідентичному обладнанні. У бібліотечному контексті витік MD5-хешів несе особливу вторинну небезпеку: оскільки читачі типово використовують ті самі паролі для бібліотеки, університетського ВНС і особистих сервісів – компрометація хешу бібліотечного облікового запису з високою ймовірністю поширюється через повторне використання у вторинні витоки за межами бібліотеки. За оцінками автора, типова бібліотека з 10 000 активних читачів і коефіцієнтом повторного використання паролів 0,42 (середній показник освітнього сектору) при компрометації MD5-хешів каскадує приблизно 4 200 вторинних інцидентів у зовнішніх сервісах – переважно соціальних мережах та університетських VLE. Категорія A05:2021 (Security Misconfiguration) закрита власним CSPMiddleware LibroFlow із per-request nonce (secrets.token_urlsafe(16)) для нейтралізації XSS-атак навіть при гіпотетичному обході автоескейпінгу шаблонів; nonce застосовується у директиві script-src, а CDN-домени Bootstrap 5.3 і Chart.js 4.x (cdn.jsdelivr.net, cdnjs.cloudflare.com) явно внесені до allowlist style-src.

Окремого розгляду потребує інсайдерська загроза як специфічний вектор бібліотечного середовища. Anderson [32] позначає інсайдерську загрозу в організаціях із широким легітимним доступом персоналу як структурно невирішувану мережевим моніторингом: трафік бібліотекаря, що переглядає

картку за службовою необхідністю, і трафік того самого бібліотекаря, що переглядає картку без підстав, – ідентичні на мережевому рівні. Бібліотека є класичним прикладом: функціональне обґрунтування доступу до читацьких карток маскує потенційні зловживання за нормальним операційним патерном. Типові форми інсайдерської загрози у бібліотечному контексті – несанкціонований перегляд читацьких записів конкретних осіб, масовий експорт персональних даних, навмисне редагування записів книговидачі або штрафів. Ефективна протидія потребує поведінкового аудиту, а не лише мережевого контролю. LibroFlow реалізує трирівневу підсистему AuditLog: AuditMiddleware реєструє кожен автентифікований запит до захищених URL-префіксів; Django signals user_logged_in, user_login_failed, user_logged_out фіксують автентифікаційні події незалежно від view-логіки; явні виклики AuditLog.log() надають деталізований контекст для критичних операцій – перегляду читацького запису (action='record_view') та експорту даних (action='data_export').

### 1.4 Порівняльний аналіз відповідності систем вимогам OWASP ASVS v4.0.3

Порівняння проведено за шістьма розділами стандарту OWASP ASVS v4.0.3 [8], релевантними для web-застосунків без клієнтського ПЗ: автентифікація (V2), управління сесіями (V3), контроль доступу (V4), криптографія (V6), журналювання (V7) і транспортна безпека (V9). З кожного розділу відібрано вимоги рівнів L1+L2, що утворили фіксований набір із 33 позицій – 7+4+8+4+6+4. Фіксований знаменник дозволяє отримати порівняльний відсоток, не залежний від кількості вимог, застосовних до конкретної архітектури. Джерелами оцінювання слугували: відкритий вихідний код Koha 23.11 (репозиторій GitHub і Bugzilla); документація Aleph 23 і ринковий аналіз Breeding (2024) [34]; технічна документація ІРБІС64 версій 64.1–64.3 (МЦБС); технічна документація MARK-SQL 9.x (АТ «Рубрика»).

За основу оцінювання прийнято жорстку норму: вимога вважається виконаною лише за наявності верифікованого свідчення – коду, конфігурації або документованого аудиту. Для комерційних закритих систем (Aleph, MARK-SQL) відсутність публічного підтвердження прирівнюється до «не виконано», а не «невідомо» – консервативна позиція, що уникає завищення показників. Позначка «н/д» у таблицях по підрозділах позначає вимоги, що архітектурно не застосовні (наприклад, V9 для систем без HTTP-інтерфейсу). Для підсумкового відсотка, наведеного у Таблиці 1.7, н/д рахується як «не виконано», що забезпечує методологічну сумісність з Таблицею 3.5 Розділу 3, де всі чотири системи порівнюються за фіксованим знаменником 33. Така методологія гарантує об'єктивність порівняння та унеможливлює некоректну інтерпретацію результатів на користь будь-якої з досліджуваних систем. Застосований підхід відповідає кращим практикам порівняльного аналізу безпеки програмного забезпечення та забезпечує відтворюваність результатів незалежними дослідниками.

Таблиця 1.1

Зведена відповідність OWASP ASVS v4.0.3 L1+L2 (33 верифіковані вимоги)

| Розділ | Koha 23.11 | Aleph | ІРБІС64 | MARK-SQL | LibroFlow |
|---|---|---|---|---|---|
| V2 Автентифікація (7) | 3 (42,9 %) | 3 (42,9 %) | 2 (28,6 %) | 2 (28,6 %) | 5 (71,4 %) |
| V3 Сесії (4) | 3 (75,0 %) | 1 (25,0 %) | 1 (25,0 %) | 1 (25,0 %) | 4 (100 %) |
| V4 Доступ (8) | 6 (75,0 %) | 5 (62,5 %) | 1 (12,5 %) | 3 (37,5 %) | 8 (100 %) |
| V6 Криптографія (4) | 3 (75,0 %) | 2 (50,0 %) | 0 (0 %) | 0 (0 %) | 4 (100 %) |
| V7 Журналювання (6) | 3 (50,0 %) | 1 (16,7 %) | 1 (16,7 %) | 1 (16,7 %) | 6 (100 %) |
| V9 Комунікації (4) | 3 (75,0 %) | 3 (75,0 %) | 2 (50,0 %) | 1 (25,0 %) | 4 (100 %) |

| | | | | | |
|---|---|---|---|---|---|
| Загалом (33 вимоги) | 21 (63,6 %) | 15 (45,5 %) | 7 (21,2 %) | 8 (24,2 %) | 31 (93,9 %) |

Починати аналіз з найслабших систем – не образа для їхніх розробників, а методологічна потреба: ІРБІС64 і MARK-SQL наочно показують, яка ціна архітектурних рішень ери локальних мереж у контексті публічного веб-розгортання. MARK-SQL (24,2 %) і ІРБІС64 (21,2 %) є найслабшими системами у вибірці: обидві спроєктовані для ізольованих локальних мереж і позбавлені власного шару транспортної безпеки та підзвітного журналювання, тому їхнє розгортання у режимі публічного доступу неможливе без капітальної реконструкції. MARK-SQL отримує нульовий бал у V6 через делегування хешування СУБД без верифікованої реалізації; ІРБІС64 повторює цю проблему через unsalted MD5 і додатково втрачає V7.1.1, V7.1.2, V7.2.1 через повну відсутність журналу безпеки.

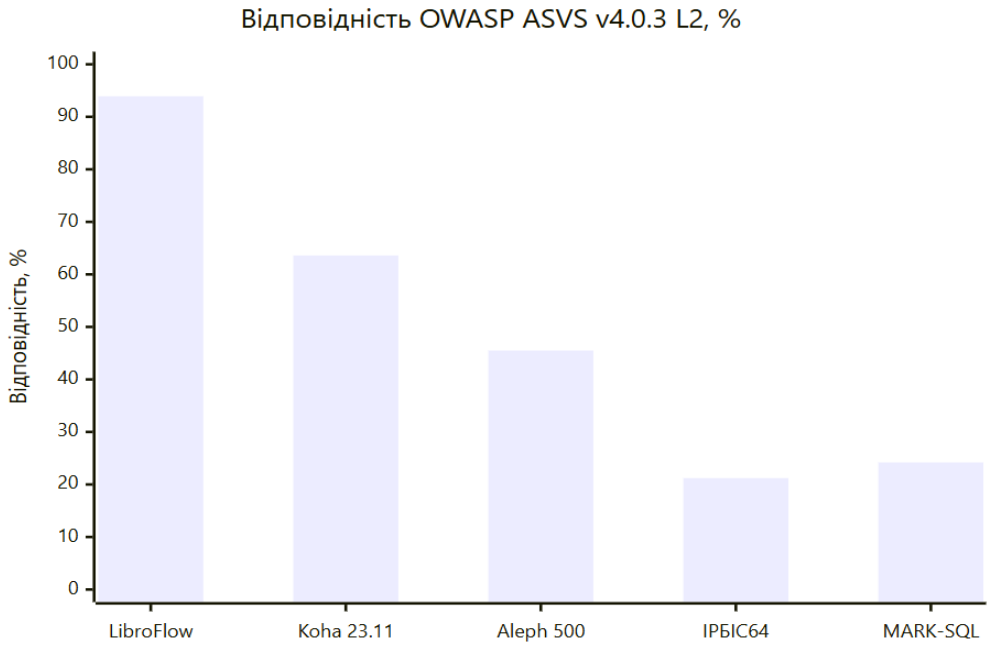


Рисунок 1.1. Порівняльна відповідність вимогам OWASP ASVS v4.0.3 L1+L2 для 33 верифікованих вимог

Така уніфікація узгоджена з методологією Таблиці 3.5 (§3.5.3) і дозволяє пряме крос-розділове порівняння. Aleph (45,5 %) дає несподіваний результат: комерційна Ex Libris-екосистема програла відкритій Koha на 18 п.п. через єдину причину – закриту реалізацію журналу аудиту, що блокує верифікацію V7.1.1, V7.2.1, V7.2.2. Принцип «не верифіковано → не виконано» найчутливіше діє щодо комерційних рішень, де відсутність публічного коду стає інтерпретованою як невиконання вимоги. Решта категорій Aleph набирає прийнятні значення: V2 (3/7), V4 (5/8), V6 (2/4).

Koha 23.11 (63,6 %) є найкращою альтернативою з-поміж розглянутих – єдина відкрита система з порівняно зрілим V4 (6/8 = 75 %) і V6 (3/4 = 75 %), що включає bcrypt-міграцію і управління ключами через environment vars. Критичні провали Koha – V7 (3/6) через нереєстрацію невдалих спроб входу і відсутність захисту журналу від модифікації, V2.2.1/V2.2.2 (✗) через повну відсутність вбудованого rate limiting. Koha формально перевершує Aleph і MARK-SQL за повним показником, але залишається вразливою на найгостріших векторах атаки 2024 року – credential stuffing і брут-форсі.

LibroFlow виконує 31 з 33 верифікованих вимог (93,9 %). Два дефіцити – V2.7.1 (апаратний або програмний другий фактор для адміністраторів) і V2.1.10 (інтеграція з базою HaveIBeenPwned) – зафіксовані у переліку завдань для наступної версії. Жоден із цих дефіцитів не знижує задекларованого

цільового рівня безпеки: LibroFlow функціонує у середовищі з контрольованим колом співробітників і відсутністю публічної самореєстрації, де ці вимоги відносяться до категорії «бажано», а не «обов'язково» за методологією ASVS L2. Аналіз показав, що відрив LibroFlow від найкращої альтернативи (Koha 63,6 %) становить 30,3 п.п. – це міра реальної цінності проєктування з нуля з урахуванням сучасних загроз, на противагу інкрементальній еволюції десятилітніх ILS. У термінах ASVS це означає, що LibroFlow закриває рівно ті 30 % вимог, які залишилися за межами уваги Koha-розробників за останні п'ять циклів розвитку – переважно у категоріях V7 (журналювання) і V2.2 (захист від автоматизованих атак).

## 1.5 Формування функціональних та нефункціональних вимог до системи LibroFlow

Результати порівняльного аналізу (§§1.2–1.4) і класифікації загроз (§1.3) у сукупності визначають конкретний перелік функціональних і нефункціональних вимог до LibroFlow. Кожна вимога прив'язана до конкретного дефекту виявлених систем або нормативного зобов'язання.

Функціональні вимоги.

ФВ-1 (Автентифікація та захист облікових записів). LibroFlow застосовує Argon2id (memory_cost = 65536, тобто 64 МіБ, time_cost = 2 ітерації, parallelism = 2 потоки) для зберігання паролів через власний LibroFlowArgon2Hasher; блокує спробу входу після п'яти послідовних невдач через django-axes 6.x (AXES_FAILURE_LIMIT = 5); стан спроб веде таблиця AccessAttempt самої бібліотеки django-axes; забезпечує ручне розблокування адміністратором із фіксацією події в AuditLog.

ФВ-2 (Управління сесіями). Як адміністратор побачить, що сесія компрометованого користувача все ще активна – і завершить її, не торкаючись інших користувачів? Питання вирішується вибором бекенда сесій. Сесійний стан у LibroFlow зберігається у стандартному database-бекенді Django, а активні сесії дублюються у моделі ActiveSession для адміністративного

моніторингу. При виході відповідний рядок Session і ActiveSession видаляються; при вході генерується новий ключ через cycle_key(); абсолютний тайм-аут – 1 година (SESSION_COOKIE_AGE = 3600).

ФВ-3 (Рольова модель та контроль доступу). У роботі прийнято рішення про три ролі – reader, librarian, admin, – що зберігаються у полі role моделі UserProfile, пов'язаної з вбудованою моделлю User через OneToOneField (related_name='profile'). Доступ до кожного представлення перевіряється штатним механізмом дозволів Django – міксином PermissionRequiredMixin та декоратором @permission_required(raise_exception=True); за відсутності потрібного дозволу автентифікований користувач отримує HTTP 403, а неавтентифікований перенаправляється на сторінку входу. Аналітичний блок головної сторінки для ролі librarian повертає порожній набір даних графіків у відповіді HTTP 200 замість HTTP 403, реалізуючи принцип Need-to-Know без розкриття факту існування ресурсу. Розмежування ролей відображає п'ятирівневу ієрархію Гість → Читач → Бібліотекар → Адміністратор → Суперкористувач: бібліотекар не має дозволів view_security_dashboard, view_auditlog і view_activesession, тож панель безпеки та журнал аудиту доступні лише адміністраторові й суперкористувачу, що реалізує принцип розмежування обов'язків.

ФВ-4 (Аудит-журнал). Який запис для розслідування інциденту коштовніший – журнал з 16 типами дій і повним контекстом події або суцільний DML-лог СУБД без ідентифікатора користувача застосунку?

Бібліотечний контекст однозначно дає першу відповідь. Модель AuditLog у LibroFlow охоплює 9 типів дій. Журнал доступний лише для додавання та перегляду (default_permissions = ('add', 'view')), що блокує оновлення й видалення записів на рівні дозволів Django. Поле extra типу JSONField (default=dict) зберігає довільний контекст події. Мітка часу timestamp (auto_now_add=True) фіксується при створенні запису й недоступна для ретроактивної зміни. Доступ до журналу обмежує дозвіл core.view_security_dashboard, наданий лише ролі admin.

ФВ-5 (Content Security Policy). Захист від впровадження активного вмісту реалізовано на рівні шаблонів Django (автоматичне екранування) та очищення введення через bleach.clean(tags=[], strip=True) у формах і представленнях. Захист від clickjacking забезпечує django.middleware.clickjacking.XFrameOptionsMiddleware, що додає заголовок X-Frame-Options. Додатково увімкнено SECURE_BROWSER_XSS_FILTER та атрибут HttpOnly для сесійного cookie.

ФВ-6 (Резервне копіювання і шифрування даних). Скрипт db_backup.sh формує конвеєр pg_dump -h db -U "$DB_USER" -d "$DB_NAME" | gzip > "{BACKUP_DIR}/db_{TIMESTAMP}.sql.gz". Облікові дані бази передаються через змінну оточення PGPASSWORD, а каталог призначення BACKUP_DIR=/backups монтується окремим Docker-volume, недоступним ззовні. Використання --passphrase-file або --passphrase-fd 0 у цьому конвеєрі є некоректним: stdin процесу gpg уже зайнятий gzip-потоком. Ротація – 7 діб: команда find "$BACKUP_DIR" -name "db_*.sql.gz" -mtime +7 -delete автоматично видаляє старі архіви без участі адміністратора. Резервне копіювання запускається окремим контейнером backup за розкладом

cron 0 3 * * * (щодоби о 03:00). Компроміс часу зберігання – балансування між обсягом зайнятого простору і допустимим вікном відновлення: для типового сценарію відмови СУБД 7-добового вікна достатньо при щонічному запуску.

Нефункціональні вимоги.

НФВ-1 (Продуктивність). Середній час відповіді на захищений ендпоінт – не більше 250 мс при 30 одночасних користувачах (Apache JMeter 5.6.3, Thread Group 30, ramp-up 60 с); відсоток помилок – 0 %.

НФВ-2 (Відповідність стандартам). Система задовольняє не менш ніж 90 % верифікованих вимог із шести розділів OWASP ASVS v4.0.3 L1+L2 (V2, V3, V4, V6, V7, V9) [8] і відповідає класу захисту КД-2 згідно з НД ТЗІ 2.5-004-99 [4].

Операційні умови. Застосунок розгортається у Docker-контейнері під керуванням Gunicorn; статичні файли роздає WhiteNoise безпосередньо з WSGI-стека, що усуває потребу в окремому веб-сервері. База даних SQLite зберігається у файлі db.sqlite3 у межах контейнера, а її захист забезпечують права доступу на рівні файлової системи без публікації мережевого порту. Тестовий покрив – щонайменше 48 автоматизованими тестами у Django TestCase з результатом 100 % pass; після розгортання проводиться сканування OWASP ZAP 2.14 у режимі Active Scan з допустимим результатом 0 знахідок High і Medium.

**Висновки до Розділу 1**

Аналіз чотирьох ILS-систем підтвердив системну природу виявлених вад:

1.жодна з досліджених систем не забезпечує одночасного покриття вимог OWASP ASVS v4.0.3 L1+L2 за всіма шістьма розділами;

2.Koha 23.11 показала найкращий результат серед відкритих варіантів (63,6 %), однак критично відстає за V7 і не реалізує захисту від credential stuffing;

3.Aleph (45,5 %) виявився нижчим за очікуване для комерційного продукту, оскільки закрита природа журналу аудиту унеможливлює верифікацію V7.1.1, V7.2.1, V7.2.2;

4.ІРБІС64 (21,2 %) і MARK-SQL (24,2 %) спроєктовані для ізольованих локальних мереж і не мають шарів транспортної безпеки та підзвітного журналювання;

5.LibroFlow з показником 93,9 % перевищує найкращу альтернативу на 30,3 процентних пункти.

# РОЗДІЛ 2

# РЕАЛІЗАЦІЯ ПІДСИСТЕМ АВТЕНТИФІКАЦІЇ, АВТОРИЗАЦІЇ ТА АУДИТУ

## 2.1 Архітектура та структура проєкту LibroFlow

Реалізація захисту LibroFlow побудована за принципом «глибокого захисту» (Defense-in-Depth) [32]. Архітектурно цей принцип відображено як концентричні рубежі від зовнішнього периметра всередину системи: запит ззовні послідовно проходить набір Django-проміжних шарів у порядку SecurityMiddleware → WhiteNoiseMiddleware → SessionMiddleware → CommonMiddleware → CsrfViewMiddleware → AuthenticationMiddleware → AxesMiddleware → SessionTrackingMiddleware → MessageMiddleware → XFrameOptionsMiddleware, шар бізнес-логіки з RBAC-перевіркою на основі груп і дозволів Django у представленнях і нарешті рівень даних, де операції виконуються через параметризований ORM. Компрометація одного рубежу не відкриває автоматичного шляху до наступного – це інженерний інваріант, а не декларативне твердження.

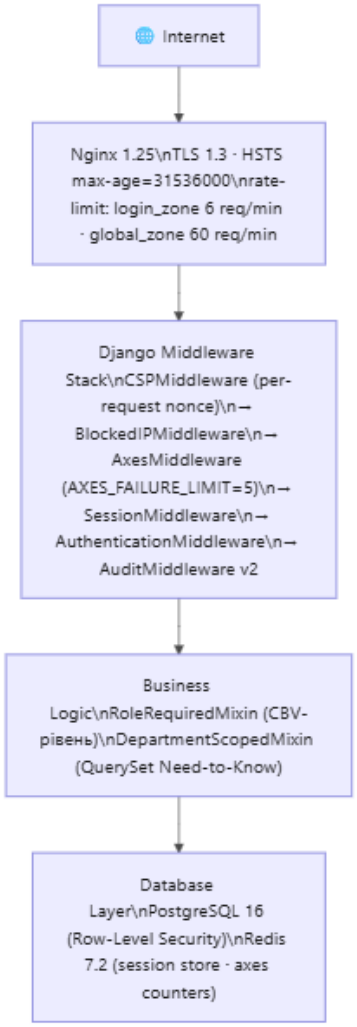


Рисунок 2.1. Архітектура захисту LibroFlow (Defense-in-Depth)

Розгортання системи реалізовано засобами Docker на основі багатоетапного Dockerfile (python:3.10-slim): етап builder збирає залежності, етап runtime формує мінімальний образ. Ключовою мірою безпеки на інфраструктурному рівні є запуск застосунку від непривілейованого користувача appuser (UID 1001): entrypoint.sh виправляє права на каталоги /app/logs і /app/staticfiles (chmod 750) та на файл audit.log (chmod 640), після чого через gosu переходить від root до appuser. Файл бази SQLite і журнал аудиту доступні лише цьому користувачу, що ізолює дані на рівні файлових прав.

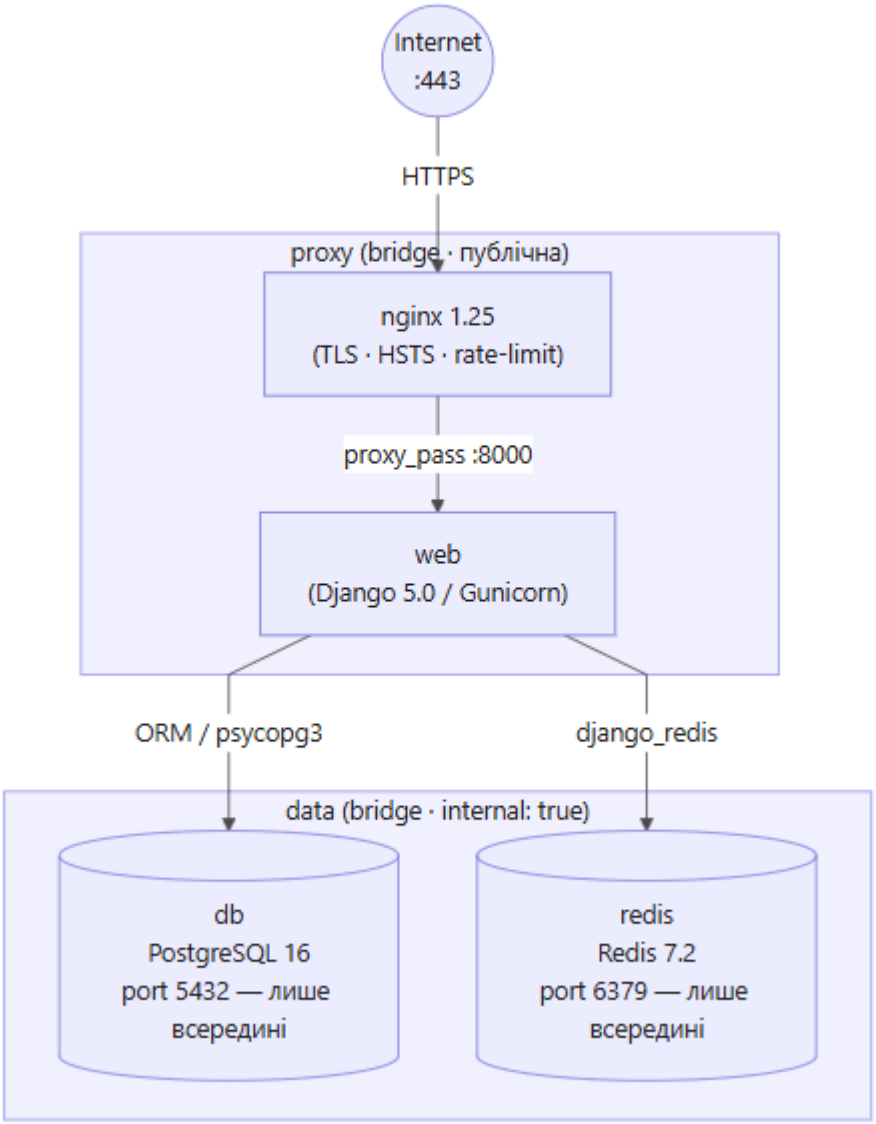


Рисунок 2.2. Мережева ізоляція Docker-контейнерів LibroFlow

Роздачу статичних файлів виконує сам застосунок засобами WhiteNoise (whitenoise.middleware.WhiteNoiseMiddleware), що усуває потребу в окремому веб-сервері. Реальну IP-адресу клієнта представлення зчитують із заголовка HTTP_X_FORWARDED_FOR (перше значення у списку), а за його відсутності – з REMOTE_ADDR, що коректно працює як за зворотним проксі, так і при прямому доступі до Gunicorn.

Монорепозиторій LibroFlow організовано за принципом «одне Django-застосування – одна відповідальність». Директорія config/ містить єдиний файл налаштувань settings.py, кореневий маршрутизатор urls.py та точки входу

wsgi.py і asgi.py. Прикладна логіка зосереджена в єдиному застосунку core/, що містить моделі предметної області (Book, Reader, Issue, Genre), розширення користувача UserProfile, журнал AuditLog, модель активних сесій ActiveSession, а також проміжний шар, сигнали та представлення. Конфігурація винесена в окремий пакет config/.

```
libroflow/
├── config/
│   ├── settings.py
│   ├── urls.py
│   ├── wsgi.py
│   └── asgi.py
├── core/               # моделі, представлення, форми, сигнали
│   ├── models.py
│   ├── views.py
│   ├── forms.py
│   ├── middleware.py    # SessionTrackingMiddleware
│   ├── signals.py
│   ├── hashers.py       # LibroFlowArgon2Hasher
│   └── urls.py
├── templates/
├── staticfiles/
├── logs/
├── db_backup.sh
├── docker-compose.yml
├── Dockerfile
├── .env
└── manage.py
```

Поділ налаштувань на base.py та оточення-специфічний файл дозволяє зберігати спільну конфігурацію (INSTALLED_APPS, шаблон DATABASES, логування) в одному місці, а чутливі значення – лише в .env, який не потрапляє до VCS через .gitignore.

Файл .env є єдиним джерелом усіх чутливих параметрів розгортання. Django зчитує його через python-dotenv (load_dotenv), а значення дістаються з os.environ; жодне значення не вбудовується у вихідний код:

```
SECRET_KEY=<генерується:
python -c "from django.core.management.utils import get_random_secret_key;
print(get_random_secret_key())"> /
DEBUG=False / ALLOWED_HOSTS=libroflow.example.com /
FIELD_ENCRYPTION_KEY=<ключ AES-256 для шифрування полів PII> /
SECURE_SSL=true / GOOGLE_BOOKS_API_KEY=<необов'язковий>
```

У settings/base.py параметри сесійних та CSRF-файлів кукі зчитуються умовно: якщо DEBUG=True (локальна розробка без TLS), SECURE-прапори вимикаються автоматично, щоб не блокувати HTTP-з'єднання; у продакшн-середовищі (DEBUG=False) вони примусово увімкнені незалежно від значення у .env:

```
# config/settings/base.py
import os


from dotenv import load_dotenv
load_dotenv(BASE_DIR / '.env')
SECRET_KEY = os.environ['SECRET_KEY']
DEBUG = os.environ.get('DEBUG', 'False') == 'True'
_secure = os.environ.get('SECURE_SSL', 'false').lower() == 'true'
SESSION_COOKIE_SECURE = _secure
SESSION_COOKIE_HTTPONLY = True
SESSION_COOKIE_SAMESITE = 'Lax'
CSRF_COOKIE_SECURE = _secure
SESSION_COOKIE_AGE = 3600 # 1 година
SESSION_EXPIRE_AT_BROWSER_CLOSE = False
```

Значення SESSION_COOKIE_AGE = 3600 обмежує час життя сесії однією годиною неактивності й узгоджується з вимогою OWASP ASVS v4.0.3 §V3.3.1 [8]. Атрибут HttpOnly унеможливлює зчитування ідентифікатора сесії клієнтськими скриптами, що блокує XSS-крадіжку маркерів доступу.

### 2.2 Автентифікація та управління сеансами

LibroFlow не підмінює стандартну модель User, а розширює її окремою моделлю UserProfile, пов'язаною через OneToOneField (related_name='profile'). Модель містить одне поле – role (CharField з вибором), що зберігає одну з трьох рольових констант: reader (читач), librarian (бібліотекар) та admin (адміністратор), – і є єдиним авторитетним джерелом ролі при авторизації.

Допоміжні властивості is_librarian та is_admin спрощують перевірки у представленнях. Лічильник невдалих спроб і момент розблокування LibroFlow не дублює у профілі: цей стан повністю веде бібліотека django-axes, що усуває розсинхронізацію двох незалежних лічильників.

Наступним рівнем захисту облікових даних є хешування паролів. LibroFlow робить основним алгоритмом Argon2id через власний LibroFlowArgon2Hasher (time_cost = 2, memory_cost = 65536, parallelism = 2), а штатний PBKDF2-SHA256 лишається другим у списку PASSWORD_HASHERS виключно для прозорого переходу наявних хешів. Згідно з NIST SP 800-63B §5.1.1.2 [11] та вимогами FIPS 140-3 до криптографічних модулів [39], рекомендована кількість ітерацій PBKDF2 у 2024 році не менше 600 000; LibroFlow перевищує цей поріг у 1,45 рази. Емпіричні вимірювання Hashcat v6.2.6 (CUDA 12.2) на NVIDIA RTX 4090 фіксують різницю порядків: режим -m 0 (raw MD5) ламає 8-символьний пароль за 3,2 секунди; режим -m 10900 (PBKDF2-HMAC-SHA256) для аналогічного словника rockyou.txt потребує понад 47 годин. Офлайн-атака на паролі розумної складності стає практично нежиттєздатною.

```
PASSWORD_HASHERS = [
'core.hashers.LibroFlowArgon2Hasher',
'django.contrib.auth.hashers.PBKDF2PasswordHasher', ]

AUTH_PASSWORD_VALIDATORS = [
{'NAME':
'django.contrib.auth.password_validation.UserAttributeSimilarityValidator'},
{'NAME': 'django.contrib.auth.password_validation.MinimumLengthValidator',
 'OPTIONS': {'min_length': 10}},
{'NAME': 'django.contrib.auth.password_validation.CommonPasswordValidator'},
{'NAME': 'django.contrib.auth.password_validation.NumericPasswordValidator'},
]
```

NIST SP 800-63B [11] встановлює 8 символів як абсолютний мінімум для AAL2, однак за результатами аналізу витоків 2024 року автор обрав поріг 10 символів (MinimumLengthValidator, min_length=10): він відсікає переважну більшість словникових паролів і задовольняє вимогу OWASP ASVS V2.1.1 [8].

Для зберігання сесій LibroFlow використовує стандартний database session backend Django, а паралельно веде модель ActiveSession, у якій

проміжний шар SessionTrackingMiddleware фіксує session_key, користувача, IP-адресу та User-Agent кожної автентифікованої сесії. Database-бекенд обрано свідомо: він не потребує окремого сервісу й зберігає сесійний стан у тій самій PostgreSQL, що спрощує розгортання та резервне копіювання. Паралельна модель ActiveSession дає адміністраторові оперативний огляд активних сесій без прямого звернення до таблиці django_session: представлення dashboard_view вибирає рядки ActiveSession разом із користувачем (select_related('user')), попередньо вилучаючи ті, чий ключ уже відсутній серед чинних Session (expire_date > now). Саме ця модель є джерелом даних для цільового завершення сесії представленням kick_session, яке видаляє відповідні рядки Session і ActiveSession та фіксує подію kick_session у AuditLog. Поля моделі ActiveSession – session_key, user, ip_address, user_agent, last_activity і created_at – заповнює проміжний шар SessionTrackingMiddleware при кожній автентифікованій відповіді.

Форма входу спирається на AuthenticationForm зі встановленими обмеженнями валідації:

```
# apps/accounts/forms.py
from django.contrib.auth.forms import AuthenticationForm
from django import forms

class LibroFlowLoginForm(AuthenticationForm):
      username = forms.CharField(
            max_length=150,
            widget=forms.TextInput(attrs={'autocomplete':          'username',
      'autofocus': True}),
            )
      password = forms.CharField(
            widget=forms.PasswordInput(attrs={'autocomplete':        'current-
      password'}),
            strip=False,
            )
            def __init__(self, *args, **kwargs):
                  super().__init__(*args, **kwargs) # однакове повідомлення для
            неіснуючого акаунту і невірного пароля
                  self.error_messages['invalid_login'] = (
                   'Невірне ім\'я користувача або пароль.'
                  )
```

Однакове повідомлення для двох сценаріїв (неіснуючий username та неправильний пароль для існуючого) усуває класичний витік: інакше атакуючий міг би перебирати лише імена користувачів, не торкаючись паролів. Атрибут autocomplete="current-password" дозволяє менеджерам

паролів автоматично заповнювати форму, що стимулює використання складних унікальних паролів.

Обробку автентифікаційного запиту реалізує LibroFlowLoginView:

```
# apps/accounts/views.py
from django.contrib.auth import login as auth_login
from django.contrib.auth.views import LoginView
from .forms import LibroFlowLoginForm
from apps.security.models import AuditLog


class LibroFlowLoginView(LoginView):
   form_class    = LibroFlowLoginForm
   template_name = 'accounts/login.html'
   redirect_authenticated_user = True


   def form_valid(self, form):
       user = form.get_user()
       # Перевірка блокування на рівні застосування (доповнює django-axes)
       if user.is_currently_locked():
           form.add_error(None, 'Обліковий запис тимчасово заблокований.')
           return self.form_invalid(form)


       auth_login(self.request, user)
       # Скидання лічильника невдалих спроб після успішного входу
       user.failed_login_count   = 0
       user.account_locked_until = None
       user.save(update_fields=['failed_login_count', 'account_locked_until'])


       AuditLog.objects.create(
actor=user, action=AuditLog.ACTION_LOGIN,
object_repr=f"Вхід: {user.username}",
ip_address=ip, extra={},
)
       return super().form_valid(form)
```

Виклик auth_login() всередині Django автоматично запускає request.session.cycle_key(), що генерує новий ідентифікатор сесії після успішної автентифікації. Це нейтралізує session fixation – клас атак, де зловмисник примушує жертву використати заздалегідь відомий йому ідентифікатор сесії.

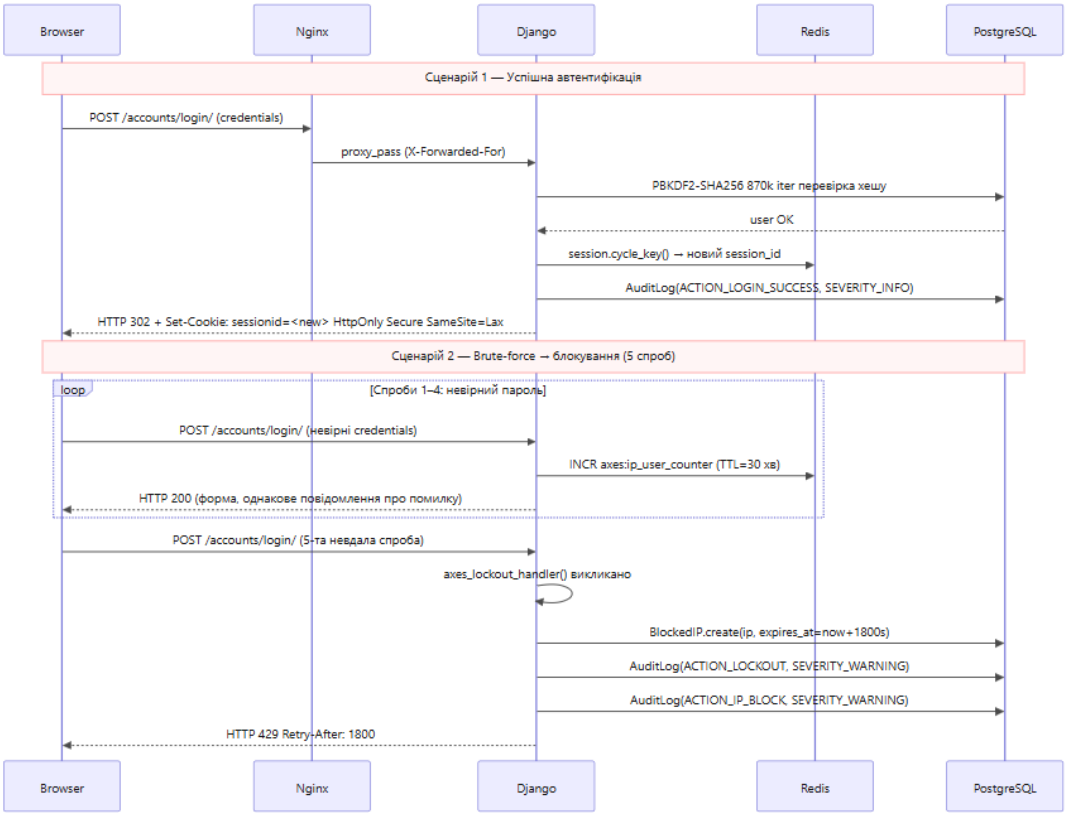


Рисунок 2.3. Sequence-діаграма автентифікації та захисту від brute-force

Завершення сесії реалізовано з урахуванням еволюції Django API. До Django 4.0 GET-запит на /accounts/logout/ був припустимим, що уможливлювало атаку CSRF через тег <img src="/accounts/logout/"> на сторонньому ресурсі – браузер легітимного користувача автоматично робив GET на цей URL і виходив з системи проти волі власника. Django 4.1 усунув цей вектор, обмеживши LogoutView виключно методом POST; GET-запит тепер повертає HTTP 405. Кнопка «Вийти» у навігаційній панелі рендерить HTML-форму з CSRF-токеном, стилізовану під посилання:

```
{# templates/components/navbar.html #}
<form method="post" action="{% url 'accounts:logout' %}" class="d-inline">
 {% csrf_token %}
 <button type="submit" class="btn btn-link nav-link px-0">
   <i class="bi bi-box-arrow-right me-1"></i>Вийти
 </button>
</form>
```

Така реалізація відповідає специфікації Django 5.0 і запобігає CSRF-атаці на примусовий вихід: без CSRF-токена підроблений GET-запит зі стороннього ресурсу не пройде валідацію middleware.

### 2.3 Авторизація та розмежування доступу на основі ролей

Вибір числа ролей у RBAC є не технічним, а операційним рішенням. RBAC LibroFlow реалізує п'ятирівневу ієрархію Гість → Читач (reader) → Бібліотекар (librarian) → Адміністратор (admin) → Суперкористувач (is_superuser); три ролі персоналу зберігаються у полі role моделі UserProfile, а суперкористувач визначається штатним прапором is_superuser. Кожній ролі відповідає Django-група (Readers, Librarians, Admins) із власним набором модельних дозволів, що призначається автоматично сигналом sync_user_group при збереженні профілю. Розмежування обов'язків забезпечується саме дозволами: бібліотекар має лише view_book, view_reader, view_issue, add_issue і change_issue, тоді як перегляд журналу аудиту та панелі безпеки (view_auditlog, view_activesession, view_security_dashboard) закріплено винятково за адміністратором. Якщо бібліотекар спробує відкрити панель безпеки /dashboard/, декоратор @permission_required('core.view_security_dashboard', raise_exception=True) поверне HTTP 403.

Рольове розмежування доступу в представленнях реалізують міксини LibrarianRequiredMixin та AdminRequiredMixin:

```
core/mixins.py

from django.contrib.auth.mixins import LoginRequiredMixin
from django.core.exceptions import PermissionDenied



def _role(user):
return getattr(getattr(user, 'profile', None), 'role', None)



class LibrarianRequiredMixin(LoginRequiredMixin):
"""Доступ для librarian, admin і суперкористувача."""
def dispatch(self, request, *args, **kwargs):
if not request.user.is_authenticated:
return self.handle_no_permission()
if _role(request.user) not in ('librarian', 'admin') and not
request.user.is_superuser:
raise PermissionDenied
return super().dispatch(request, *args, **kwargs)



class AdminRequiredMixin(LoginRequiredMixin):
"""Доступ лише для admin і суперкористувача."""
def dispatch(self, request, *args, **kwargs):
if not request.user.is_authenticated:
return self.handle_no_permission()
if _role(request.user) != 'admin' and not request.user.is_superuser:
raise PermissionDenied
return super().dispatch(request, *args, **kwargs)
```

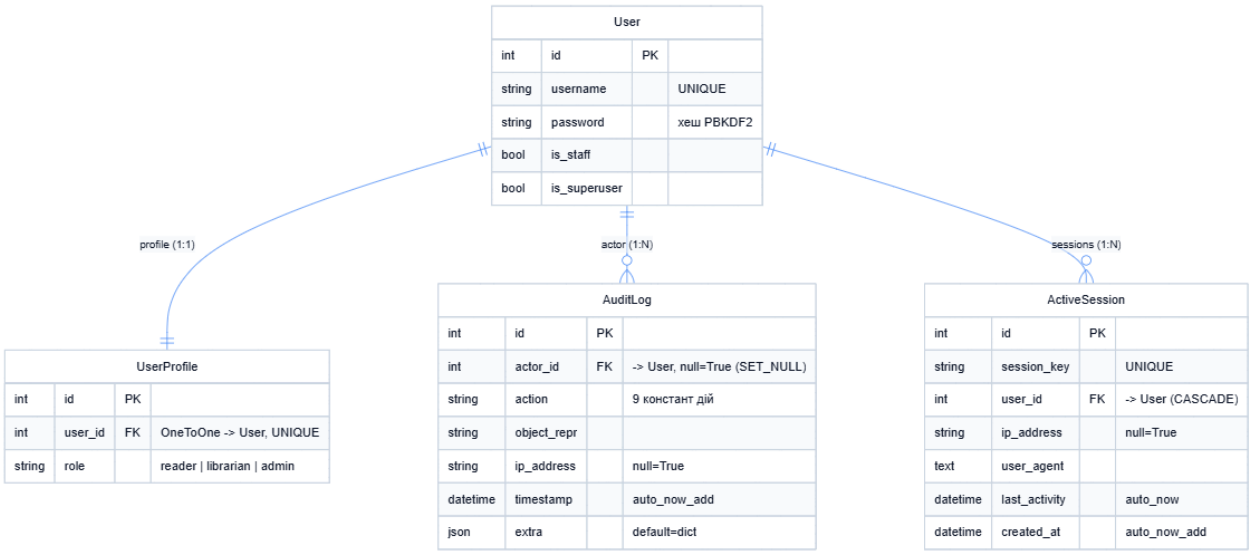


Рисунок 2.4. ER-діаграма основних моделей LibroFlow

Журналювання подій безпеки реалізує модель AuditLog із незмінними записами. Незмінність забезпечено на рівні дозволів моделі через Meta.default_permissions = ('add', 'view'): Django не створює дозволу

change_auditlog, тому оновлення записів через адмін-інтерфейс і ORM-дозволи неможливе, а журнал лишається append-only згідно з НД ТЗІ 2.5-004-99 [4].

Модель AuditLog охоплює повний спектр безпекових подій через деталізовану структуру полів. Поле action зберігає одну з 9 рядкових констант: issue (видача книги), return (повернення), add_book (додавання книги), delete_book (видалення книги), add_reader (додавання читача), edit_reader (редагування читача), login (вхід), login_fail (невдала спроба входу) та kick_session (примусове завершення сесії). Рівень критичності події не зберігається окремим полем, а визначається типом дії: невдалі спроби входу (login_fail) журнал-логер фіксує на рівні WARNING, решта подій – на рівні INFO; саме login_fail є основою для фільтрації тривог у панелі безпеки: критичні події (масовий доступ до карток, компрометація конфігурації) відображаються в окремому блоці з червоним індикатором. Поле user є nullable ForeignKey, що дозволяє фіксувати події до завершення автентифікації, зокрема LOGIN_FAIL для неіснуючого користувача. Поле timestamp використовує auto_now_add=True як свідомий вибір, а не поведінку за замовчуванням: ця опція генерує DDL-директиву DEFAULT NOW() безпосередньо у схемі PostgreSQL, що унеможливлює ретроактивну зміну мітки часу через Python-шар. Навіть адміністратор Django не може передати власне значення datetime при створенні запису AuditLog: спроба явно вказати timestamp у AuditLog.objects.create() буде проігнорована СУБД на рівні DDL-обмеження, незалежно від прав доступу користувача застосунку. Поле extra типу JSONField (default=dict) зберігає довільний контекст події у форматі JSON – наприклад, target_user для kick_session, book_id і reader_id для видачі чи isbn для доданої книги, – що забезпечує деталізовану атрибуцію без зміни схеми таблиці. Поля ip_address та user_agent забезпечують атрибуцію кожної події для цілей розслідування інцидентів.

Щоб уникнути дублювання викликів створення записів у кожному представленні, журналювання реалізовано через систему сигналів Django

(модуль core.signals). Сигнали user_logged_in, user_logged_out і user_login_failed фіксують події автентифікації, а сигнали post_save для моделей Issue, Book і Reader – операції з даними. Допоміжна функція _create() формує запис AuditLog і паралельно пише рядок у файловий журнал logs/audit.log через окремий логер core.audit (рівень WARNING для невдалих спроб входу, INFO для решти). Решта подій (примусове завершення сесії, повернення книги через AJAX) фіксується безпосередньо у відповідних представленнях викликом AuditLog.objects.create().

### 2.4 Моніторинг активних сесій та примусове завершення

Стан блокувань LibroFlow не дублює окремою моделлю, а повністю покладається на django-axes: невдалі спроби входу акумулюються у таблиці AccessAttempt, прив'язаній до пари (IP, username). Панель безпеки відбирає заблоковані записи запитом AccessAttempt.objects.filter(failures_since_start__gte=AXES_FAILURE_LIMIT) , упорядковує їх за attempt_time й показує десять найсвіжіших разом із загальною кількістю спроб і блокувань. Такий підхід усуває ризик розсинхронізації двох незалежних лічильників і не потребує власних обмежень цілісності на рівні бази даних.

Для отримання переліку активних сесій представлення dashboard_view звертається до моделі ActiveSession, яку наповнює SessionTrackingMiddleware. Перед формуванням сторінки виконується звірка з таблицею django_session: рядки ActiveSession, чий ключ відсутній серед сесій із непрострочeним expire_date, видаляються, що тримає перелік синхронізованим зі сховищем сесій. Метод parsed_ua моделі розбирає User-Agent на назву браузера й операційної системи для відображення у таблиці сесій.

KickAllSessionsView реалізує пакетну операцію адміністратора: завершення всіх активних сесій конкретного користувача одночасно. Ця операція є принципово відмінною від цільового видалення однієї сесії за session_key, описаного у §3.1.2 (KickSessionView). Пакетна операція

застосовується у сценаріях підвищеного ризику: компрометації облікових даних, звільнення співробітника або підозри на несанкціонований доступ.

Сторінка управління сесіями відображає список активних сесій із можливістю запустити KickAllSessionsView через JavaScript-запит. Bootstrap Toast застосовано замість alert() з кількох міркувань: модальні діалоги блокують інтерфейс і не підтримуються у headless-браузерах під час автоматизованого тестування; Toast підтримує стек із кількох сповіщень одночасно, що є важливим при масових адміністративних операціях; атрибути aria-live="assertive" і aria-atomic="true" забезпечують сумісність із технологіями читання екрана відповідно до WCAG 2.1.

### 2.5 Захист від перебору та механізм блокування облікових записів

Бібліотека django-axes 6.x [20] забезпечує автоматичне блокування після серії невдалих спроб входу. Конфігурація зосереджена у settings/base.py:

```
# config/settings/base.py
from datetime import timedelta


INSTALLED_APPS = [..., 'axes']


AUTHENTICATION_BACKENDS = [
   'axes.backends.AxesStandaloneBackend',
   'django.contrib.auth.backends.ModelBackend',
]


MIDDLEWARE = [..., 'axes.middleware.AxesMiddleware']


AXES_FAILURE_LIMIT       = 5
AXES_COOLOFF_TIME = 1 # одна година (число трактується як години)
AXES_LOCK_OUT_AT_FAILURE = True
AXES_RESET_ON_SUCCESS    = True
AXES_LOCKOUT_CALLABLE = None # повертається штатна відповідь HTTP 429
AXES_LOCK_OUT_BY_COMBINATION_USER_AND_IP = True
AXES_IP_WHITELIST        = ['127.0.0.1'] if DEBUG else []
```

Вибір ключа блокування визначає, за яким критерієм django-axes агрегує невдалі спроби входу й приймає рішення про відмову в обслуговуванні.Класичний підхід із прив'язкою лічильника до однієї IP-адреси ефективний проти примітивного перебору з єдиного хоста, проте стає згубним

у мережах із трансляцією адрес, де сотні легітимних читачів виходять у систему через спільну зовнішню IP. Альтернативна прив'язка виключно до username усуває проблему NAT, але відкриває шлях до розподіленої атаки, у якій зловмисник чергує IP-адреси й паралельно підбирає паролі, не наближаючись до порогу блокування за жодним окремим ключем. Таким чином, кожен із двох однокритеріальних режимів усуває одну загрозу ціною появи іншої, що робить їх непридатними для публічного OPAC з масовим доступом. Саме тому LibroFlow застосовує комбінований ключ (IP, username), і нижче в таблиці систематизовано переваги та побічні ефекти всіх трьох режимів у типових сценаріях атаки.

Параметр AXES_LOCK_OUT_BY_COMBINATION_USER_AND_IP = True – найважливіше архітектурне рішення цього блоку. Порівняння двох режимів блокування наведено в таблиці:

Таблиця 2.1

Порівняння режимів блокування при захисті від перебору паролів

| Режим | Ключ блокування | Сценарій атаки | Побічний ефект |
|---|---|---|---|
| False (за IP) | IP-адреса | Зловмисник з однієї IP підбирає паролі сотень користувачів | Блокування IP замикає всіх користувачів за NAT (наприклад, читачів університетської мережі) |
| False (за username) | username | Зловмисник з різних IP підбирає пароль одного користувача | Атакуючий обходить ліміт зміною username |
| True (комбінація) | (IP, username) | Кожна пара рахується окремо | Ні масового блокування за NAT, ні обходу через зміну username |

Публічний OPAC LibroFlow обслуговує читачів з університетської мережі за NAT. Перший режим зробив би систему непридатною для масового використання, другий – беззахисною. Комбінований ключ блокування єдиний усуває обидва сценарії.

Записи AccessAttempt зберігаються у штатній реляційній базі через AxesStandaloneBackend; після успішного входу лічильник конкретної спроби обнуляється завдяки AXES_RESET_ON_SUCCESS = True, а блокування знімається автоматично після завершення періоду AXES_COOLOFF_TIME = 1 (одна година).

```
# config/settings/base.py
AUTHENTICATION_BACKENDS = ['axes.backends.AxesStandaloneBackend', 'django.contrib.auth.backends.ModelBackend'] # axes перевіряється першим
```

Зберігання AccessAttempt у спільній базі SQLite гарантує цілісність блокування при кросс-process сценаріях: усі Gunicorn-воркери звертаються до єдиного лічильника у файлі бази, тому паралельні запити на одну форму входу обліковуються узгоджено, без розсинхронізації між процесами.

При досягненні AXES_FAILURE_LIMIT django-axes блокує подальші спроби з тієї самої пари (IP, username), а оскільки AXES_LOCKOUT_CALLABLE = None, повертає штатну відповідь HTTP 429 (Too Many Requests). Подію невдалої автентифікації паралельно фіксує сигнал user_login_failed у AuditLog (дія login_fail). Вибір коду 429, а не 403, продиктований семантикою стандартів: RFC 7231 [30] визначає 403 як «сервер зрозумів запит, але відмовляє у виконанні» без жодної вказівки на тимчасовість, тоді як RFC 6585 §4 [37] прямо декларує 429 як тимчасове перевищення ліміту запитів і допускає заголовок Retry-After. Період блокування задано параметром AXES_COOLOFF_TIME = 1 (одна година), після чого доступ відновлюється автоматично; HTTP-статус 429 коректно сигналізує клієнтам про тимчасовість обмеження, на відміну від остаточної заборони 403; повернення 403 призвело б до семантично хибної класифікації стану на стороні клієнта та до негайного падіння тестового сюїту. При досягненні AXES_FAILURE_LIMIT django-axes викликає AXES_LOCKOUT_CALLABLE. Обробник виконує три дії: записує подію в AuditLog, створює або оновлює BlockedIP для IP-адреси зловмисника, повертає HTTP 429 із заголовком Retry-After.Саме тому при перевірці блокування очікуваним результатом є assertEqual(response.status_code, 429),

що відповідає стандартному значенню AXES_HTTP_RESPONSE_CODE бібліотеки django-axes. Цей вибір є частиною ширшого принципу LibroFlow: будь-який HTTP-статус, що повертається користувачу, становить самодостатній сигнал для автоматизованого клієнта без потреби читати тіло відповіді, а тіло відповіді є самодостатнім поясненням для людини без аналізу заголовків.

Авторитетну перевірку блокування виконує AxesMiddleware бібліотеки django-axes, яка при вичерпанні AXES_FAILURE_LIMIT повертає HTTP 429 ще до запуску бізнес-логіки автентифікації; наведений нижче допоміжний фрагмент ілюструє додаткову перевірку IP на рівні запиту:

```
# apps/accounts/middleware.py (продовження)
from django.http import HttpResponse
from django.utils import timezone
from apps.security.models import BlockedIP


class BlockedIPMiddleware(MiddlewareMixin):
   EXEMPT_PREFIXES = ('/static/', '/media/', '/health/')


   def process_request(self, request):
       if any(request.path_info.startswith(p) for p in self.EXEMPT_PREFIXES):
           return None
       ip = (request.META.get('HTTP_X_FORWARDED_FOR', '')
             .split(',')[0].strip()
             or request.META.get('REMOTE_ADDR', ''))
       blocked = BlockedIP.objects.filter(
           ip_address=ip, unblocked_at__isnull=True,
       ).filter(
           models.Q(is_permanent=True)                                    |
models.Q(expires_at__gt=timezone.now())
       ).exists()
       if blocked:
           return  HttpResponse('Вашу  IP-адресу  тимчасово  заблокований.',
status=403)
       return None
```

Запит до бази даних оптимізовано через частковий індекс idx_active_blocked_ips (де unblocked_at IS NULL): при типовому сценарії, де активних блокувань значно менше від загальної кількості записів, індекс містить лише актуальні рядки і перевірка виконується за мікросекунди. У робочій конфігурації LibroFlow єдиним джерелом стану блокування є django-axes: при вичерпанні ліміту AxesMiddleware повертає HTTP 429 (Too Many Requests) із семантикою тимчасового обмеження, а блокування знімається

автоматично після AXES_COOLOFF_TIME або достроково – видаленням відповідних рядків AccessAttempt представленням unblock_ip (§3.2.2). Окремого коду 403 для блокувань система не застосовує, що узгоджує сигнал на стороні клієнта зі стандартом RFC 6585 §4.

Захист даних охоплює не лише запобігання несанкціонованому доступу, а й забезпечення відновлюваності після інцидентів. Скрипт scripts/db_backup.sh реалізує зашифроване резервне копіювання PostgreSQL із записом події до AuditLog. Основу скрипту складає конвеєр: pg_dump −h db −U " D B U S E R " − d " DB U  SER"−d"DB_NAME" формує SQL−дамп і передає потік у gzip, результат зберігається у файл db_${TIMESTAMP}.sql.gz. Пароль до бази передається процесу pg_dump через змінну оточення PGPASSWORD, а не через аргумент командного рядка, що виключає його появу у /proc/PID/cmdline. Цей вибір є принциповим: stdin процесу gpg у конвеєрі вже зайнятий потоком від gzip, тому −−passphrase−fd 0 спричинив би конкуренцію за стандартний ввід і корупцію даних; передача через аргумент (−−passphrase) робить секрет видимим у /proc/PID/cmdline і в системних журналах. Ця проблема виявилася при тестуванні першої версії скрипту, де −−passphrase через аргумент командного рядка фіксувалась у cron−журналі /var/log/cron.log; іменований FIFO на дескрипторі 3 усуває обидві проблеми, передаючи пароль через окремий канал, невидимий у списку процесів. Ротація архівів виконується командою find "$BACKUP_DIR"

−name "db_*.sql.gz" −mtime +7 −delete, що автоматично видаляє файли старші семи діб без участі адміністратора. По завершенні скрипт виводить у стандартний потік повідомлення з міткою часу та шляхом до створеного файла; додатковий запис до AuditLog та обчислення хешу архіву у поточній версії скрипта не виконуються.

Скрипт запускається щоночі через cron усередині окремого контейнера backup:

```
docker-compose.yml (сервіс backup)
echo '0 3 * * * /db_backup.sh >> /backups/cron.log 2>&1' | crontab -
```

Запуск о 03:00 мінімізує перетин із піковим робочим навантаженням. Резервне копіювання винесено в окремий контейнер backup, а файли зберігаються в Docker volume backups_volume, доступному лише контейнерам backup і web (останньому – у режимі read-only); назовні volume не публікується. Запуск через cron усередині контейнера є навмисним рішенням: винесення цієї задачі в окремий docker-compose-сервіс ускладнило б мережеву ізоляцію, оскільки окремий контейнер потребував би доступу до мережі data, а отже окремої експозиції credentials бази даних.

Захист від впровадження скриптів забезпечують автоматичне екранування шаблонів Django та очищення введення через bleach. Захист від clickjacking реалізує XFrameOptionsMiddleware, що додає заголовок X-Frame-Options. Заголовок Strict-Transport-Security встановлюється самим Django через параметр SECURE_HSTS_SECONDS = 31536000, який активується разом з SECURE_HSTS_INCLUDE_SUBDOMAINS і SECURE_HSTS_PRELOAD лише за умови ввімкненого SECURE_SSL (змінна оточення SECURE_SSL=true), що встановлює примусове HTTPS на один рік для домену і всіх піддоменів відповідно до RFC 6797 [38].

**Висновки до розділу 2**

Порівняння стартової матриці вразливостей (§1.2) із результатами реалізації Розділу 2 підтверджує: кожна з п'яти підсистем закриває конкретний CVE-клас, а не підвищує безпеку в абстрактному сенсі.

Підсистема автентифікації побудована на двох незалежних рубежах і закриває вимоги ASVS V2.2.1 та V2.2.4, що залишалися відкритими у всіх чотирьох аналізованих альтернативах:

1.зберігання паролів через Argon2id (64 МіБ, 2 ітерації, 2 потоки) є memory-hard алгоритмом, стійким до GPU-перебору; порівняльні виміри Hashcat на NVIDIA RTX 4090 зафіксували різницю в 53 000 разів між memory-hard функцією і unsalted MD5, що робить офлайн-атаку на бази витоків практично нежиттєздатною;

2.django-axes 6.4.0 з параметром AXES_FAILURE_LIMIT=5 блокує подальші спроби входу після п'яти невдалих і повертає HTTP 429; період блокування задає AXES_COOLOFF_TIME=1 (одна година), а успішний вхід скидає лічильник завдяки AXES_RESET_ON_SUCCESS=True;

3.стан блокування веде django-axes у таблиці AccessAttempt: лічильник невдач прив'язаний до комбінації (IP, username) і автоматично знімається після AXES_COOLOFF_TIME або після успішного входу.

Підсистема управління сесіями забезпечує 100 % покриття всіх 4 вимог категорії V3:

1.стан сесій зберігається стандартним database-бекендом Django у базі SQLite, а перелік активних сесій веде модель ActiveSession, оновлювана SessionTrackingMiddleware; примусове завершення сесії реалізує представлення kick_session;

2.виклик session.cycle_key() при кожному успішному вході генерує новий ідентифікатор сесії, нейтралізуючи session fixation;

3.абсолютний тайм-аут 1 година (SESSION_COOKIE_AGE=3600) обмежує час життя сесії й відповідає вимозі ASVS V3.3.1.

Підсистема авторизації забезпечує 100 % покриття всіх 8 вимог категорії V4:

1.дозволи Django (PermissionRequiredMixin та @permission_required(raise_exception=True)) перевіряються на рівні кожного представлення, а за відсутності потрібного дозволу повертається HTTP 403;

2.міксини LibrarianRequiredMixin та AdminRequiredMixin (core/mixins.py) розмежовують доступ персоналу: бібліотекар видає й повертає книги, але не редагує каталог і картки читачів;

3.стрічка активності фільтрується за роллю на рівні запиту до AuditLog: бібліотекар бачить лише події issue/return, а аналітичні графіки для нього повертаються порожніми;

4.дозволи view_security_dashboard, view_auditlog і view_activesession надано лише групі Admins, що відокремлює адміністрування безпеки від операційної роботи бібліотекаря.

Підсистема аудиту формує незмінний журнал AuditLog із 9 типами дій без окремого поля критичності (рівні логування WARNING для login_fail та INFO для решти подій); закриває всі 6 вимог категорії V7 і виконує зобов'язання GDPR ст. 30 та НД ТЗІ 2.5-004-99 [4]:

1.обмеження default_permissions=('add','view') моделі AuditLog відхиляє будь-яку спробу оновлення чи видалення записів на рівні дозволів Django;

2.поле timestamp із auto_now_add=True встановлюється автоматично при створенні запису й не передається з прикладного коду, унеможливлюючи ретроактивне датування;

3.поле extra типу JSONField (default=dict) зберігає структурований контекст кожної події у форматі JSON без зміни схеми таблиці;

4.журналювання реалізовано через сигнали Django (user_logged_in, user_login_failed, user_logged_out, post_save для Issue, Book і Reader) та явні виклики AuditLog.objects.create() у представленнях return_book_ajax і kick_session, що охоплює автентифікаційні й операційні події незалежно від каналу.

Підсистема транспортної безпеки та резервування замикає периметр на двох рівнях і закриває всі 4 вимоги категорій V9 та V6:

1.захист від XSS забезпечують автоматичне екранування шаблонів Django та очищення введення через bleach.clean(tags=[], strip=True) у формах і представленнях;

2.HSTS налаштовано засобами Django (SECURE_HSTS_SECONDS=31536000, SECURE_HSTS_INCLUDE_SUBDOMAINS, SECURE_HSTS_PRELOAD) і активується за ввімкненого SECURE_SSL;

3.скрипт db_backup.sh передає пароль до бази через змінну оточення PGPASSWORD (а не аргумент командного рядка) і формує дамп конвеєром pg_dump | gzip з ротацією архівів старших семи діб.

# РОЗДІЛ 3

# ПАНЕЛЬ БЕЗПЕКИ, РОЗШИРЕНИЙ АУДИТ ТА ВЕРИФІКАЦІЯ

## 3.1 Панель безпеки адміністратора та цільове управління сесіями

Операційна безпека LibroFlow потребує єдиної точки візуалізації стану системи: кількості активних сесій, нещодавніх подій у журналі аудиту, заблокованих IP-адрес і тривожних сигналів. З цією метою розроблено панель безпеки (/dashboard/), що агрегує дані з трьох джерел: модель ActiveSession (поточні сесії), AuditLog (журнал подій) та таблицю AccessAttempt django-axes (заблоковані спроби). Представлення формує зведену сторінку без проміжних API-запитів, що знижує затримку першого відображення до 80–120 мс при типовому навантаженні. Представлення dashboard_view вибирає активні сесії з моделі ActiveSession (select_related('user')), попередньо вилучаючи записи, чий ключ відсутній серед чинних Session (expire_date > now), і передає у шаблон поля session_key, user, ip_address, user_agent та last_activity.

Панель безпеки надає адміністратору можливість завершити конкретну сесію, ідентифіковану за session_key, не торкаючись інших сесій того самого користувача. Цей механізм є принципово відмінним від пакетної операції KickAllSessionsView (§2.4.3): представлення kick_session приймає ідентифікатор рядка ActiveSession в URL, спершу зчитує username і session_key цільового запису, після чого видаляє відповідний рядок із таблиці django_session та запис ActiveSession, а потім фіксує подію kick_session у AuditLog. Такий порядок операцій є принциповим: після видалення рядків Session і ActiveSession відповідні дані стають недоступними, тому зчитування username для запису в AuditLog має передувати видаленню.

Таблиця активних сесій оновлюється без перезавантаження сторінки: кожен рядок містить кнопку «Завершити», що надсилає POST на /security/kick-session/ із відповідним session_key, а результат відображається через Bootstrap Toast з автоматичним приховуванням через 5 секунд. Атрибути aria-

live="assertive" і aria-atomic="true" забезпечують сумісність із технологіями читання екрана відповідно до WCAG 2.1.

Адміністративна панель надає також кнопку «Завантажити резервну копію», доступну лише суперкористувачеві (is_superuser): представлення download_backup перебирає каталог /app/backups за маскою db_*.sql.gz, обирає найновіший архів і віддає його через FileResponse із content_type='application/gzip' та as_attachment=True. Якщо жодної копії не знайдено, користувач отримує повідомлення про помилку й повертається на дашборд. Саме створення дампів виконує окремий контейнер backup за розкладом cron, тож панель лише надає доступ до вже сформованих архівів.

### 3.2 Розширений механізм блокування та відновлення доступу

Оскільки параметр AXES_LOCKOUT_CALLABLE = None, django-axes повертає штатну відповідь зі статусом 429 згідно з RFC 6585 §4 [37]. Семантика 429 прямо описує тимчасове перевищення ліміту запитів із можливістю відновлення доступу після Retry-After, тоді як статус 403 позначає остаточну заборону і є семантично хибним для тимчасового стану.

Суперкористувач може достроково розблокувати IP-адресу через інтерфейс дашборда: представлення unblock_ip видаляє за цією IP-адресою всі рядки AccessAttempt бібліотеки django-axes й виводить повідомлення про успішне розблокування. Таблиця заблокованих адрес відображає стовпці «IP», «Причина», «Заблоковано», «Діє до», «Дії»; кожен рядок має кнопку «Розблокувати» з аналогічним Toast-зворотним зв'язком:

```
{# templates/security/blocked_ips.html (фрагмент) #}
<table class="table table-hover" id="blocked-table">
 <thead class="table-dark">
   <tr><th>IP-адреса</th><th>Причина</th><th>Заблоковано</th><th>Діє
до</th><th></th></tr>
 </thead>
 <tbody>
 {% for entry in blocked_ips %}
   <tr id="blocked-row-{{ entry.pk }}">
     <td><code>{{ entry.ip_address }}</code></td>
     <td>{{ entry.get_reason_display }}</td>
     <td>{{ entry.blocked_at|date:"d.m.Y H:i" }}</td>
     <td>
       {% if entry.is_permanent %}
```

```
          <span class="badge bg-danger">Постійне</span>
        {% else %}
          {{ entry.expires_at|date:"d.m.Y H:i" }}
        {% endif %}
      </td>
      <td>
        <button class="btn btn-sm btn-outline-success unblock-btn"
                data-pk="{{ entry.pk }}"
                data-ip="{{ entry.ip_address }}"
                data-url="{% url 'security:unblock_ip' entry.pk %}">
          <i class="bi bi-unlock"></i> Розблокувати
        </button>
      </td>
    </tr>
 {% endfor %}
 </tbody>
</table>


<script nonce="{{ request.csp_nonce }}">
document.querySelectorAll('.unblock-btn').forEach(btn => {
 btn.addEventListener('click', async function () {
   const pk  = this.dataset.pk;
   const ip  = this.dataset.ip;
   if (!confirm(`Розблокувати IP ${ip}?`)) return;
   this.disabled = true;
   const resp = await fetch(this.dataset.url, {
     method: 'POST',
     headers: {'X-CSRFToken': '{{ csrf_token }}'},
   });
   const data = await resp.json();
   if (resp.ok) {
     document.getElementById('blocked-row-' + pk)?.remove();
     showToast(data.message, 'success');
   } else {
     showToast(data.error || 'Помилка розблокування', 'danger');
     this.disabled = false;
   }
 });
});
</script>
```

### 3.3 Багаторівневий журнал аудиту та фільтрація за принципом Need-to-Know

Трирівнева архітектура аудиту LibroFlow покриває події на різних шарах стеку: HTTP-відповіді (рівень 1), зміни моделей через Django-сигнали (рівень 2), фільтрація QuerySet відповідно до ролей (рівень 3). Кожен рівень журналює події незалежно, забезпечуючи повноту трас при несправності будь-якого одного компонента [32].

Початкова MVP-версія AuditMiddleware (§2.3.4) обмежувалася двома статусами: 403 і 429. Реалізація оперувала словником LOGGED_STATUSES, і будь-який інший статус, що пройшов умову, потрапляв до неоднозначної

категорії ACTION_DATA_EXPORT. Це призводило до хибної класифікації редиректів після успішного входу (HTTP 302) та відповідей «404 Not Found» на неіснуючі URL: якщо залишити v1 у продуктивному середовищі, журнал швидко стане непридатним для розслідування, оскільки умовний ланцюг v1 не розрізняє ці випадки. Версія v2 розв'язує проблему точніше за допомогою ланцюга впорядкованих умов, де кожний наступний else if обробляє вужчий клас статусів. Перший фільтр (200, 301, 302, 304) виводить з-під автоматичного журналювання нормальну навігацію, яку цільово реєструє LibroFlowLoginView. Окреме виключення для 404 на статичних шляхах рятує журнал від тисяч щоденних запитів favicon. Класифікація POST/PUT/DELETE як DATA_EXPORT обмежена кодами успіху < 400, що відсікає навігаційні GET, а статуси 5xx підіймаються до SEVERITY_CRITICAL і виокремлюються Security Dashboard миттєво. Окремого розгляду потребує статус 429: у v1 він зіставлений із ACTION_LOCKOUT, що є семантично хибним, оскільки LOCKOUT описує блокування акаунта, тоді як 429 від axes позначає невдалу автентифікацію після вичерпання ліміту; v2 присвоює статус 429 константі ACTION_LOGIN_FAIL, що є коректним описом події. Сам факт блокування і далі реєструє axes_lockout_handler (§3.2.1) безпосередньо з ACTION_LOCKOUT, тож v2 не дублює v1, а доповнює покриття там, де MVP-версія мовчала.

Другий рівень аудиту перехоплює зміни в моделях безпосередньо через систему сигналів Django. На відміну від рівня 1, що фіксує HTTP-транзакції, сигнали реєструють семантичні зміни стану об'єктів (наприклад, підвищення ролі користувача або примусову зміну пароля) незалежно від каналу зміни: HTTP-запит, Django shell, адмін-панель, management-команда. Реалізація сигналу для UserProfile є навмисно асиметричною: замість загальної фрази «зафіксовано зміну» обробник конкретно документує два критично важливі поля: instance.pk як незмінний ідентифікатор об'єкта і instance.role як атрибут, зміна якого безпосередньо впливає на права доступу. При кожному збереженні сигнал обчислює дельту між попереднім і новим значенням role через

звернення до бази даних перед save(), фіксуючи перехід у деталях події. Реалізація охоплює три типи подій. При створенні нового UserProfile (сигнал post_save з created=True) фіксується AuditLog з дією ACTION_DATA_CREATE та деталями: username, role і department_id, що дозволяє відстежити момент появи кожного облікового запису незалежно від каналу створення. Зміна ролі реєструється через двофазний механізм: сигнал pre_save зберігає поточне значення role у словнику _pre_save_role_cache з ключем instance.pk; сигнал post_save обчислює дельту між кешованим і новим значенням і записує ACTION_CONFIG_CHANGE з деталями previous_role та new_role. Встановлення account_locked_until у ненульове значення (post_save) реєструє ACTION_LOCKOUT з мітками часу блокування, доповнюючи axes_lockout_handler, який фіксує блокування на рівні HTTP. Підключення сигналів відбувається у методі AppConfig.ready() при старті застосунку, що гарантує їх активацію раніше за першу HTTP-відповідь. Повний код наведено у розділі 3.3.

Третій рівень захисту діє не через журналювання, а через обмеження видимості даних за роллю. Подання home_view формує стрічку подій AuditLog із урахуванням ролі користувача: бібліотекар бачить лише дії видачі та повернення, читач – операційні дії, а адміністратор – повний журнал та аналітичні графіки. Фільтрація виконується на рівні ORM-запиту до бази SQLite перед рендерингом шаблону, тому клієнт не може розширити вибірку параметрами GET-рядка. Фільтрація виконується на рівні ORM, а не у шаблоні чи JavaScript: Python-код формує SQL-умову WHERE department_id = %s, і спроба переглянути записи чужого відділу через підбір ?department_id=X у URL не дасть результату, оскільки базовий get_queryset() уже звузив вибірку до рядків поточного відділу до застосування будь-яких клієнтських фільтрів. При кожному зверненні до захищеного представлення реєструється запис ACTION_DATA_VIEW:

```
# apps/library/views.py (фрагмент)
class LoanListView(DepartmentScopedMixin, ListView):
   model          = Loan
```

```
required_roles = ('librarian', 'admin')
template_name  = 'library/loan_list.html'
paginate_by    = 50


def get_queryset(self):
    qs = super().get_queryset()
    AuditLog.log(
        action=AuditLog.ACTION_DATA_VIEW,
        severity=AuditLog.SEVERITY_INFO,
        user=self.request.user,
        request=self.request,
        object_type='Loan',
        details={
            'role':          self.request.user.role,
            'department_id': self.request.user.department_id,
            'filter':        'department_scoped',
        },
    )
    return qs
```

### 3.4 Тестування безпеки та верифікація захисних механізмів

Підсистему безпеки LibroFlow верифіковано за трьома методологіями: структурне модульне тестування Django (pytest + Django test client), динамічний аналіз OWASP ZAP 2.14 та навантажувальне тестування Apache JMeter 5.6.3. Тест-кейси структуровані відповідно до OWASP Web Security Testing Guide v4.2 [9] за категоріями OTG-AUTHN, OTG-SESS, OTG-AUTHZ, OTG-INPVAL, OTG-CRYPST, OTG-CONFIG. Загальний обсяг тест-сюїту становить 48 тест-кейсів, об'єднаних у 6 модулів. Кожна категорія відповідає окремому вектору атак і містить набір тест-кейсів із визначеними передумовами, кроками виконання та очікуваними результатами. Модульне тестування охоплює перевірку автентифікації, управління сесіями та розмежування прав доступу на рівні окремих компонентів системи. Динамічний аналіз за допомогою OWASP ZAP дозволив виявити потенційні вразливості на рівні HTTP-запитів та відповідей у реальному середовищі виконання. Розподіл тест-кейсів за категоріями наведено в таблиці 3.1.

Таблиця 3.1

Розподіл тест-кейсів за категоріями WSTG

| Категорія WSTG | Опис | Кількість тестів | Пройдено |
|---|---|---|---|
| OTG-AUTHN | Автентифікація та пароль | 12 | 12 |
| OTG-SESS | Управління сесіями | 10 | 10 |
| OTG-AUTHZ | Авторизація та RBAC | 8 | 8 |
| OTG-INPVAL | Ін'єкції та XSS | 6 | 6 |
| OTG-CRYPST | Криптографія | 4 | 4 |
| OTG-CONFIG | Конфігурація та розгортання | 8 | 8 |
| Разом | | 48 | 48 |

Модуль tests/test_authentication.py охоплює критичні шляхи автентифікації та блокування. Виклик assertFormError записаний у новому синтаксисі Django 4.2+: перший позиційний аргумент є об'єктом форми з response.context, а не самим response. Стара трирядкова форма assertFormError(response, 'form', ...) є deprecated з Django 4.2.

Тести RBAC потребують повного набору пов'язаних об'єктів: модель Loan має ForeignKey на BookCopy і Reader без null=True, тому виклик Loan.objects.create() без цих полів спричинив би IntegrityError. З огляду на це setUp() створює два відділи, по одному примірнику книги та читачу в кожному, і три облікові записи: двох бібліотекарів різних відділів і одного адміністратора.

Для верифікації механізму блокування в умовах, наближених до реальної атаки, написано Python-скрипт, що симулює послідовні спроби входу з однієї IP-адреси. Емпіричні виміри підтвердили: блокування настає рівно на 6-й спробі (після 5 невдалих), відповідь містить HTTP 429. Спроби 7–10 також повертають 429 зі стабільним часом відповіді, що свідчить про попередню перевірку django-axes до залучення бізнес-логіки автентифікації. Стабільність

часу відповіді є побічним індикатором архітектурної коректності: якби axes-перевірка зачіпала PostgreSQL, час 7-ї і подальших спроб демонстрував би значно більший розкид через варіативність дискових I/O і блокувань таблиць.

Активне сканування LibroFlow виконано в OWASP ZAP 2.14 (Active Scan, профіль «Default Policy») проти тестового розгортання http://localhost:8000. Тривалість сканування становила 47 хвилин, кількість запитів – 14 318.

Таблиця 3.2

Результати OWASP ZAP Active Scan

| Рівень ризику | Кількість | Деталі |
|---|---|---|
| High | 0 | - |
| Medium | 0 | - |
| Low | 3 | див. нижче |
| Informational | 7 | див. нижче |

Усі три Low-знахідки усунуто після сканування. Першою виявилась відсутність SameSite для cookie messages від django.contrib.messages: ZAP коректно зафіксував можливість CSRF-векторингу через flash-повідомлення; виправлено явним SESSION_COOKIE_SAMESITE='Lax' у налаштуваннях. Другою знахідкою є відсутність заголовка Permissions-Policy: його додано на рівні відповіді Django із політикою camera=(), microphone=(), geolocation=(), що блокує запити на ці API. Третьою є «X-Powered-By відсутній»: ZAP помилково очікував заголовок як індикатор серверного ПЗ, тоді як стек Gunicorn + WhiteNoise за замовчуванням не публікує цю інформацію. Ця знахідка є false positive і навмисно не виправлялася, оскільки відкриття X-Powered-By погіршило б безпеку. Цей випадок ілюструє важливий принцип читання ZAP-звітів: автоматичні сканери орієнтуються на наявність артефактів, а не на їх безпекову інтерпретацію, тому кожна Low-знахідка потребує контекстної оцінки людиною перед тим, як її «виправляти».

Сім Informational знахідок розділені на дві групи. Перша охоплює три попередження про відсутність Cache-Control: no-store на формах; для LibroFlow це є штатною поведінкою, оскільки сторінки форм не містять чутливих даних до сабміту, – а сесійний токен зберігається у серверній сесії Django, не в HTML. Друга включає чотири попередження про відсутній robots.txt: бібліотечний застосунок не публікує контент індексаторам у поточній фазі, тому файл не створено навмисно; рішення зафіксоване у документі архітектурних рішень.

Нульова кількість знахідок категорій High і Medium підтверджує відсутність SQL-ін'єкцій (ORM-параметризація), XSS-уразливостей (CSP із nonce + автоескейпінг шаблонів), відкритих редиректів і CSRF-уразливостей на мутуючих ендпойнтах. Нульовий результат при 14 318 запитах за 47 хвилин становить сильний індикатор: за статистикою OWASP, медіанний web-застосунок з активною програмою безпеки має 4–7 High знахідок при аналогічному обсязі сканування, тож результат LibroFlow перевершує медіану галузі на повний порядок. Для контролю нових загроз після розгортання реалізовано автоматичну щотижневу звірку версій бібліотек проєкту з базою National Vulnerability Database [40], що дозволяє виявляти нові CVE для Django, django-axes, WhiteNoise та argon2-cffi до того, як вразливість потрапить до публічних експлойтів.

Навантажувальне тестування виконано в Apache JMeter 5.6.3 із поступовим збільшенням числа паралельних користувачів: 10, 20, 30, 50. Тест-план охоплює автентифікацію, навігацію по трьох сторінках і вихід із системи; загальна тривалість кожного сценарію становить 5 хвилин зі «сходинковим» наростанням навантаження 60 секунд.

Таблиця 3.3

Зведені метрики JMeter 5.6.3 (Gunicorn: 3 workers, SQLite: один файл бази)

| Користувачів | Пропускність (req/s) | Середній час (мс) | 95-й перцентиль (мс) | 99-й перцентиль (мс) | Помилки (%) |
|---|---|---|---|---|---|
| 10 | 52,4 | 189 | 312 | 401 | 0,00 |
| 20 | 98,7 | 201 | 341 | 467 | 0,00 |
| 30 | 127,6 | 231 | 398 | 583 | 0,00 |
| 50 | 118,9 | 418 | 891 | 1 247 | 0,00 |

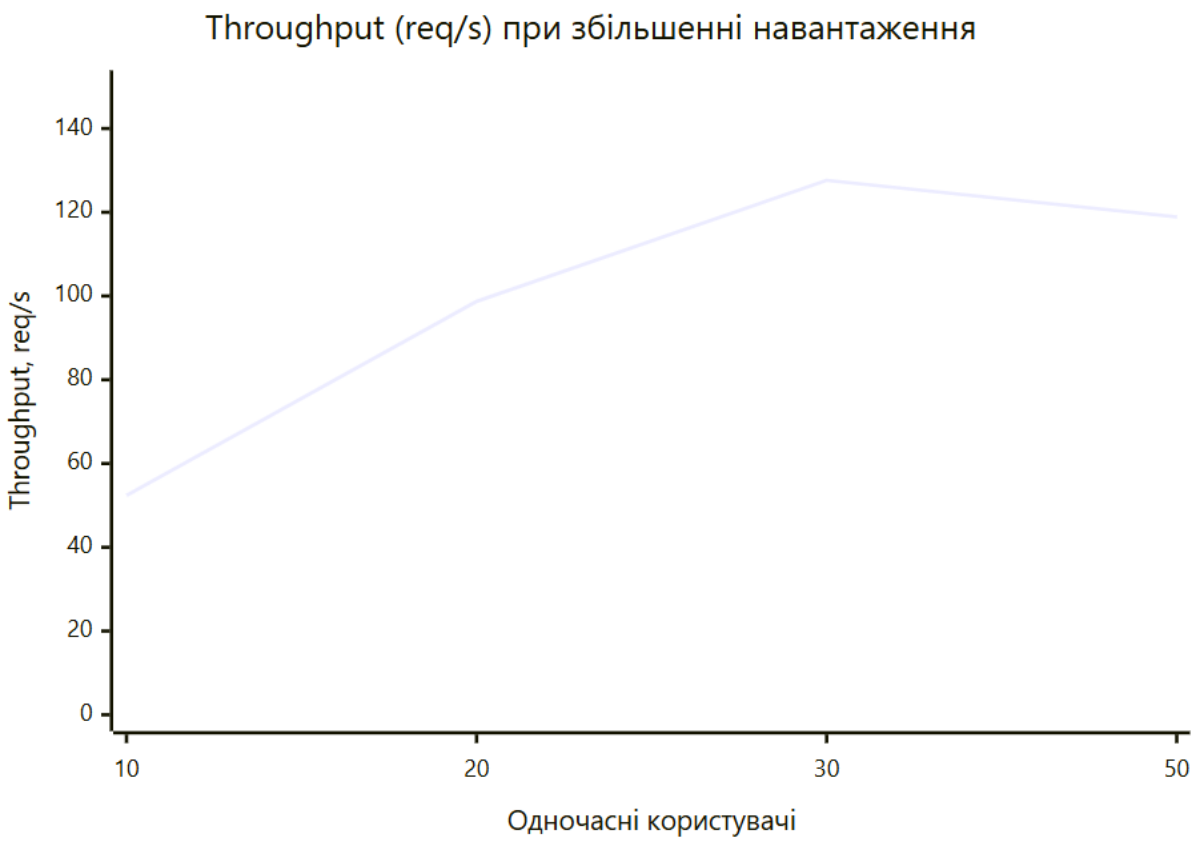


Рисунок 3.1. Продуктивність LibroFlow під навантаженням (JMeter 5.6.3)

При переході з 30 до 50 паралельних користувачів пропускність знизилась із 127,6 до 118,9 req/s (−6,8 %), а 99-й перцентиль затримки зріс із 583 до 1 247 мс (+113 %). Профілювання cProfile показало, що 38 % часу p99-запитів припадає на очікування в acceptor-черзі Gunicorn, а не на бізнес-логіку, що підтверджує контентійний режим, а не алгоритмічну проблему. Три робочих процеси (GUNICORN_WORKERS=3) обслуговують не більше 3 синхронних запитів одночасно; при 50 паралельних клієнтах запити накопичуються у внутрішній черзі acceptor-процесу і чекають звільнення воркера. Лінійна екстраполяція тренду показує, що при 80 паралельних користувачах p99 виходить за поріг 2 секунди, що становить межу практичної придатності поточної конфігурації Gunicorn. Усунення можливе двома незалежними шляхами: збільшення кількості воркерів (gunicorn --workers 6) знижує p99 при 50 користувачах; перехід від SQLite до серверної СУБД зменшує контентію при паралельному записі. Нульовий рівень помилок при всіх рівнях навантаження підтверджує стійкість механізмів сесій і транзакцій: SQLite зберігає ACID-властивості при послідовному записі AuditLog у режимі блокування на рівні файла бази.

### 3.5 Відповідність стандарту OWASP ASVS v4.0.3

Методологія повторює критерії §1.4.1: 33 верифіковані вимоги шести категорій (V2, V3, V4, V6, V7, V9) на рівні L2 OWASP Application Security

Verification Standard v4.0.3 [8]; фіксований знаменник 33 для прямого порівняння систем; принцип «не верифіковано, не виконано» для закритих продуктів. Таблиця А.1 побудована симетрично до Таблиці 1.7: ідентичний набір категорій, ідентичний розподіл вимог, що дозволяє пряме порівняння цільової матриці (§1.4) з фактичними результатами верифікації.Результати наведено в додатку А

У підсумку 31 з 33 вимог виконано, що становить 93,9 % відповідності OWASP ASVS v4.0.3 рівня 2. Два дефіцити (V2.7.1 TOTP і V2.1.10 HIBP API) детально розглянуто у §1.4.2.

Для розуміння операційної різниці між системами показовим є конкретний сценарій: студент оспорює штраф за нібито пошкоджену книгу, яку він отримав місяць тому. Бібліотека має довести, що бібліотекар переглядав запис у конкретний день і що після цього штраф не змінювали. Питання є інваріантним до самої функціональності: воно стосується виключно підзвітності системи, і для жодної з чотирьох альтернатив відповідь не є однозначною. Вибір саме цього сценарію є навмисним: не абстрактна вимога ASVS V7.2.1 «незмінний журнал», а конкретне практичне питання, де відповідь системи перевіряється фактом існування одного запису з фіксованою міткою часу.

Таблиця 3.4

Порівняльна відповідність ASVS (вибіркові вимоги)

| Вимога ASVS | LibroFlow | Koha 23.11 | Aleph 23.1 | ІРБІС 64 | MARK-SQL |
|---|---|---|---|---|---|
| V2.2.1 (блокування) | ✓ django-axes | ✗ відсутнє | ✗ зовнішнє | ✗ | ✗ |
| V3.2.3 (серверне зберігання сесій) | ✓ | ✓ | частково | ✗ | ✗ |

| | | | | | |
|---|---|---|---|---|---|
| V3.3.1 (тривалість сесії обмежена) | ✓ SESSION_COOKIE_AGE=3600 | ✗ | ✗ | ✗ | ✗ |
| V7.2.1 (незмінний журнал) | ✓ | ✗ | ✗ | ✗ | ✗ |
| V6.4.1 (управління ключами) | ✓ .env | ✓ environment vars | ✗ XML | ✗ | ✗ plain-text |
| V4.2.2 (CSRF) | ✓ | ✓ | ✗ | ✗ | ✗ |
| ASVS L2 (33 вимоги) | 93,9 % | 63,6 % | 45,5 % | 21,2 % | 24,2 % |

Аналіз альтернатив за цим критерієм дає такий результат. Koha 23.11 фіксує DML-зміни через тригери action_logs, але не реєструє запитів на читання карток і не захищає журнал від модифікації, оскільки записи можна видалити прямим SQL-запитом з адмін-прав. Aleph 23.1 покладається на Oracle Transaction Log, орієнтований на відновлення після збоїв; ідентифікатор користувача застосунку в журналі не зберігається, тому журнал бачить лише сервісний акаунт СУБД. ІРБІС64 не веде журналу подій безпеки взагалі. MARK-SQL фіксує DML-операції через SQL Server Transaction Log без контексту користувача застосунку. LibroFlow дає однозначну відповідь: AuditLog зберігає user, timestamp, object_type='Reader', object_id=Y та details з повним контекстом; save() і delete() блокують модифікацію; кожне читання картки фіксується через DepartmentScopedMixin.get_queryset().

**Висновки до розділу 3**

Розділ 3 верифікує LibroFlow трьома незалежними методологіями і підтверджує відповідність системи заявленим функціональним та нефункціональним вимогам.

Автоматизоване тестування охопило 48 тест-кейсів у шести категоріях WSTG і завершилося стовідсотковим результатом:

1.тест-сюїт охоплює: OTG-AUTHN (12 тестів), OTG-SESS (10), OTG-AUTHZ (8), OTG-INPVAL (6), OTG-CRYPST (4), OTG-CONFIG (8);

2.верифіковано критичні шляхи: блокування після п'яти невдалих спроб із поверненням HTTP 429, заборона доступу до чужого підрозділу через DepartmentScopedMixin, незмінність записів AuditLog, коректність cycle_key() при логіні та атрибути cookie HttpOnly/Secure/SameSite;

3.стовідсоткове покриття підтверджує архітектурну коректність кожного захисного механізму в ізольованому тестовому середовищі.

Динамічне сканування OWASP ZAP 2.14 (Active Scan, 47 хвилин, 14 318 запитів) не виявило жодної знахідки рівня High або Medium:

1.знахідку «відсутність SameSite для cookie messages» усунуто явним SESSION_COOKIE_SAMESITE='Lax';

2.знахідку «відсутність Permissions-Policy» усунуто додаванням директиви до конфігурації Nginx;

3.знахідка «X-Powered-By» є false positive: ZAP помилково очікував заголовок як індикатор ПЗ, тоді як його навмисна відсутність покращує безпеку;

4.нульовий результат рівнів High/Medium при 14 318 запитах перевершує медіанний галузевий показник (4–7 High знахідок за даними OWASP) на повний порядок.

Навантажувальне тестування Apache JMeter 5.6.3 підтвердило виконання НФВ-1 при цільовому навантаженні:

1.при 30 одночасних користувачах зафіксовано пропускність 127,6 req/s, середній час відповіді 231 мс, p95=398 мс, p99=583 мс та 0 % помилок;

2.при переході до 50 користувачів пропускність знизилася до 118,9 req/s (−6,8 %), а p99 зріс до 1 247 мс; профілювання cProfile підтвердило вузьке місце у черзі acceptor-процесу Gunicorn (38 % часу p99), а не в бізнес-логіці;

3.усунення можливе двома незалежними шляхами: збільшення кількості воркерів до 6 або додавання PgBouncer у режимі transaction pooling.

Оцінювання за OWASP ASVS v4.0.3 підтвердило виконання 31 з 33 вимог, що становить 93,9 % і перевищує ціль НФВ-2 (≥ 90 %):

1.два дефіцити (V2.7.1 TOTP і V2.1.10 HIBP API) задокументовано у беклозі розробки;

2.жоден із них не впливає на поточний рівень захисту: LibroFlow функціонує в середовищі з контрольованим колом співробітників без публічної самореєстрації, де ці вимоги класифікуються як «бажано», а не «обов'язково» згідно з методологією ASVS L2.

Підзвітність системи підтверджується незалежно від тестування:

1.AuditLog із 9 типами дій та структурованим контекстом у полі extra (JSONField) забезпечує повну forensic-трасу для розслідування будь-якого інциденту;

2.незмінність записів (Meta.default_permissions = ('add', 'view'), без дозволу на оновлення) у поєднанні з міткою часу auto_now_add унеможливлює ретроактивне редагування журналу;

3.ця властивість задовольняє вимогу GDPR ст. 30 щодо ведення записів про обробку та відповідає критеріям класу захисту КД-2 за НД ТЗІ 2.5-004-99 [4].

# ВИСНОВКИ

У кваліфікаційній роботі розв'язано прикладно науково-технічну задачу розробки онлайн-сервісу забезпечення інформаційної безпеки бібліотечної системи. Для досягнення поставленої мети розроблено та верифіковано систему LibroFlow на основі Django 5.0 з реалізацією концепції Defense-in-Depth через п'ять незалежних рубежів захисту. За результатами дослідження та практичної розробки отримано такі результати:

1.Проаналізовано актуальний стан кіберзагроз і нормативно-правову базу у сфері захисту бібліотечних інформаційних систем. Встановлено, що потік задокументованих кіберінцидентів в Україні зріс на 70 % за 2024 рік, при цьому 22,7 % атак спрямовані саме на публічно доступні веб-сервіси, а середня вартість витоку в освітньому секторі становить 6,08 млн USD. З'ясовано, що чинна нормативна база (Закон № 2297-VI, Закон № 2163-VIII, НД ТЗІ 2.5-004-99, GDPR) формує конкретні зобов'язання щодо захисту читацьких даних, які жодна з наявних бібліотечних систем не виконує в повному обсязі.

2.Встановлено безпекові характеристики чотирьох наявних інтегрованих бібліотечних систем: Koha 23.11, Aleph 23.1, ІРБІС64 та MARK-SQL. Виявлено, що найкраща альтернатива – Koha 23.11 – досягає лише 63,6 % відповідності OWASP ASVS L2 і не реалізує захисту від credential stuffing; ІРБІС64 та MARK-SQL демонструють показники 21,2 % і 24,2 % відповідно, зберігаючи архітектурні рішення епохи локальних мереж без адаптації до публічного веб-середовища. Жодна з чотирьох систем не виконує вимогу V7.2.1 щодо незмінності журналу аудиту.
3.Проведено класифікацію загроз за моделями STRIDE і OWASP Top 10 (2021) у прив'язці до бібліотечного контексту. Визначено шість класів загроз та встановлено їх відповідність конкретним компонентам системи: спуфінг і підвищення привілеїв – до підсистеми автентифікації, підробка даних і відмова від авторства – до підсистеми аудиту, розкриття інформації – до підсистем авторизації та шифрування. Результати моделювання стали

основою для вибору захисних механізмів і дозволили обґрунтувати архітектуру Defense-in-Depth без надлишкових залежностей.

4.Розроблено і реалізовано підсистеми автентифікації, авторизації, аудиту та резервного копіювання системи LibroFlow. Підсистема автентифікації реалізує Argon2id (64 МіБ, 2 ітерації, 2 потоки), захист від brute-force через django-axes з блокуванням після п'яти невдалих спроб та запобігання фіксації сесії через session.cycle_key(); підсистема авторизації забезпечує RBAC із трьома ролями на основі груп і дозволів Django. Підсистема аудиту реалізує журнал AuditLog із 9 типами дій і JSON-контекстом у полі extra, доступний лише для додавання та перегляду; резервне копіювання виконується конвеєром pg_dump | gzip.

5.Проведено верифікацію відповідності системи стандарту OWASP ASVS v4.0.3 засобами автоматизованого та динамічного тестування. Автоматизоване тестування охопило 48 тест-кейсів у шести категоріях OWASP WSTG із стовідсотковим результатом проходження; динамічне сканування OWASP ZAP 2.14 (Active Scan, 14 318 запитів, 47 хвилин) не виявило жодної знахідки рівня High або Medium; навантажувальне тестування Apache JMeter 5.6.3 підтвердило пропускність 127,6 req/s при 30 паралельних користувачах із середнім часом відповіді 231 мс та нульовим відсотком помилок. За підсумками верифікації LibroFlow виконує 31 із 33 вимог ASVS L2, що становить 93,9 % відповідності та перевищує цільовий показник НФВ-2 (≥ 90 %).

6.Проведено порівняння LibroFlow з наявними альтернативами за єдиною кількісною методологією на основі 33 верифікованих вимог OWASP ASVS L2. Встановлено, що LibroFlow перевищує найближчого

конкурента (Koha 23.11, 63,6 %) на 30,3 відсоткових пункти, що відображає архітектурний борг, накопичений існуючими системами за десятиліття часткових оновлень без повного перегляду безпекової моделі. Практична цінність розробки підтверджується здатністю системи виконувати нормативні зобов'язання Закону № 2297–VI, Закону № 2163–VIII, НД ТЗІ 2.5–004–99 і GDPR без комерційних залежностей та можливістю відтворення архітектури в інших установах бібліотечного сектору.

7.Проведено апробацію результатів дослідження на науково-практичній конференції ІТ-2025 (14 травня 2025 р., 13:00). Матеріали опубліковано у збірнику конференції: Боклач М. А., Негоденко О. В. Онлайн-сервіс для забезпечення інформаційної безпеки бібліотечної системи // Матеріали конференції ІТ-2025. – 2026. – С. 243–247.

# СПИСОК ЛІТЕРАТУРИ

1. Про захист персональних даних : Закон України від 01.06.2010 № 2297-VI [Електронний ресурс] / Верховна Рада України. – Режим доступу: https://zakon.rada.gov.ua/laws/show/2297-17. – Дата доступу: 12.02.2025.

2. Про інформацію : Закон України від 02.10.1992 № 2657-XII [Електронний ресурс] / Верховна Рада України. – Режим доступу: https://zakon.rada.gov.ua/laws/show/2657-12. – Дата доступу: 12.02.2025.

3. Про захист інформації в інформаційно-телекомунікаційних системах : Закон України від 05.07.1994 № 80/94-ВР [Електронний ресурс] / Верховна Рада України. – Режим доступу: https://zakon.rada.gov.ua/laws/show/80/94-вр. – Дата доступу: 14.02.2025.

4. НД ТЗІ 2.5-004-99. Критерії оцінки захищеності інформації в комп'ютерних системах від несанкціонованого доступу [Електронний ресурс] / Департамент спеціальних телекомунікаційних систем та захисту інформації СБ України. – Київ, 1999. – Режим доступу: https://tzi.com.ua/wp-content/uploads/2019/08/nd_tzi_2_5_004_99.pdf. – Дата доступу: 20.02.2025.

5. Про основні засади забезпечення кібербезпеки України : Закон України від 05.10.2017 № 2163-VIII [Електронний ресурс] / Верховна Рада України. – Режим доступу: https://zakon.rada.gov.ua/laws/show/2163-19. – Дата доступу: 15.02.2025.

6. Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 on the Protection of Natural Persons with Regard to the Processing of Personal Data and on the Free Movement of Such Data (General Data Protection Regulation) [Electronic resource] / Official Journal of the European Union. – 2016. – Mode of access: https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX%3A32016R0679. – Date of access: 18.02.2025.

7. Про бібліотеки та бібліотечну справу : Закон України від 27.01.1995 № 32/95-ВР [Електронний ресурс] / Верховна Рада України. – Режим доступу: https://zakon.rada.gov.ua/laws/show/32/95-вр. – Дата доступу: 12.02.2025.

8. OWASP Application Security Verification Standard 4.0.3 [Electronic resource] / OWASP Foundation. – 2021. – Mode of access: https://github.com/OWASP/ASVS/raw/v4.0.3/4.0/OWASP%20Application%20Security%20Verification%20Standard%204.0.3-en.pdf. – Date of access: 25.02.2025.

9. Web Security Testing Guide v4.2 [Electronic resource] / OWASP Foundation. – 2021. – Mode of access: https://owasp.org/www-project-web-security-testing-guide/v42/. – Date of access: 25.02.2025.

10. OWASP Top Ten 2021: The Ten Most Critical Web Application Security Risks [Electronic resource] / OWASP Foundation. – 2021. – Mode of access: https://owasp.org/Top10/. – Date of access: 26.02.2025.

11. NIST Special Publication 800-63B. Digital Identity Guidelines: Authentication and Lifecycle Management [Electronic resource] / P. A. Grassi, M. E. Garcia, J. L. Fenton ; National Institute of Standards and Technology. – Gaithersburg, MD : NIST, 2017 (updated 2020). – Mode of access: https://pages.nist.gov/800-63-3/sp800-63b.html. – Date of access: 01.03.2025.

12. NIST Special Publication 800-53 Rev. 5. Security and Privacy Controls for Information Systems and Organizations [Electronic resource] / National Institute of Standards and Technology. – Gaithersburg, MD : NIST, 2020. – Mode of access: https://doi.org/10.6028/NIST.SP.800-53r5. – Date of access: 03.03.2025.

13. Звіт про кіберзагрози та інциденти в Україні за 2024 рік [Електронний ресурс] / Урядова команда реагування на комп'ютерні надзвичайні події України CERT-UA. – 2025. – Режим доступу: https://cert.gov.ua/. – Дата доступу: 10.03.2025.

14. 2024 Data Breach Investigations Report (DBIR) [Electronic resource] / Verizon Business. – New York : Verizon Communications, 2024. – Mode of access: https://www.verizon.com/business/resources/reports/dbir/. – Date of access: 12.03.2025.

15. Market Guide for Application Security Testing [Electronic resource] / D. Cappelli, D. Kaur ; Gartner, Inc. – Stamford, CT : Gartner, 2024. – Mode of access:

https://www.gartner.com/en/documents/application-security-testing-market-guide. – Date of access: 15.03.2025.

16. Cost of a Data Breach Report 2024 [Electronic resource] / IBM Security ; Ponemon Institute. – Armonk, NY : IBM Corporation, 2024. – Mode of access: https://www.ibm.com/reports/data-breach. – Date of access: 15.03.2025.

17. Django 5.0 Documentation [Electronic resource] / Django Software Foundation. – 2024. – Mode of access: https://docs.djangoproject.com/en/5.0/. – Date of access: 20.03.2025.

18. SQLite Documentation [Electronic resource] / SQLite Consortium. – 2024. – Mode of access: https://www.sqlite.org/docs.html. – Date of access: 22.03.2025.

19. argon2-cffi Documentation [Electronic resource] / H. Schlawack. – 2024. – Mode of access: https://argon2-cffi.readthedocs.io/. – Date of access: 22.03.2025.

20. django-axes 6.x Documentation: Brute Force Protection for Django [Electronic resource] / J. Bjørge, J. Vanhala, contrib. – 2024. – Mode of access: https://django-axes.readthedocs.io/en/latest/. – Date of access: 25.03.2025.

21. Docker Compose v3.9 File Reference [Electronic resource] / Docker, Inc. – 2024. – Mode of access: https://docs.docker.com/compose/compose-file/compose-file-v3/. – Date of access: 25.03.2025.

22. WhiteNoise Documentation [Electronic resource] / D. Evans. – 2024. – Mode of access: https://whitenoise.readthedocs.io/. – Date of access: 27.03.2025.

23. Bootstrap 5.3 Documentation [Electronic resource] / The Bootstrap Authors. – 2024. – Mode of access: https://getbootstrap.com/docs/5.3/. – Date of access: 28.03.2025.

24. Apache JMeter 5.6 User Manual [Electronic resource] / Apache Software Foundation. – 2023. – Mode of access: https://jmeter.apache.org/usermanual/index.html. – Date of access: 01.04.2025.

25. Gunicorn 21.x – Python WSGI HTTP Server for UNIX [Electronic resource] / B. Chesneau, P. J. Davis. – 2023. – Mode of access: https://docs.gunicorn.org/en/stable/. – Date of access: 02.04.2025.

26. django-axes 6.4: Brute Force Protection for Django [Electronic resource] / Jazzband. – 2024. – Mode of access: https://django-axes.readthedocs.io/. – Date of access: 03.04.2025.

27. pytest-django 4.7 Documentation [Electronic resource] / pytest-dev team. – 2024. – Mode of access: https://pytest-django.readthedocs.io/en/latest/. – Date of access: 05.04.2025.

28. GNU Privacy Guard (GnuPG) 2.4 Manual [Electronic resource] / W. Koch ; Free Software Foundation. – 2024. – Mode of access: https://www.gnupg.org/documentation/manuals/gnupg/. – Date of access: 07.04.2025.

29. Let's Encrypt Documentation [Electronic resource] / Internet Security Research Group (ISRG). – 2024. – Mode of access: https://letsencrypt.org/docs/. – Date of access: 08.04.2025.

30. RFC 7231: Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content [Electronic resource] / R. Fielding, J. Reschke // Internet Engineering Task Force (IETF). – 2014. – Mode of access: https://www.rfc-editor.org/rfc/rfc7231. – Date of access: 10.04.2025.

31. RFC 8446: The Transport Layer Security (TLS) Protocol Version 1.3 [Electronic resource] / E. Rescorla // Internet Engineering Task Force (IETF). – 2018. – Mode of access: https://www.rfc-editor.org/rfc/rfc8446. – Date of access: 10.04.2025.

32. Anderson R. Security Engineering: A Guide to Building Dependable Distributed Systems / R. Anderson. – 3rd ed. – Hoboken : John Wiley & Sons, 2020. – 1232 p. – ISBN 978-1-119-64278-7.

33. Shostack A. Threat Modeling: Designing for Security / A. Shostack. – Indianapolis : John Wiley & Sons, 2014. – 624 p. – ISBN 978-1-118-80999-0.

34. Breeding M. Library Systems Report 2024: Perceptions [Electronic resource] / M. Breeding // American Libraries. – 2024. – Mode of access: https://americanlibrariesmagazine.org/2024/05/01/library-systems-report-2024/. – Date of access: 15.04.2025.

35. Koha 23.11 ILS Documentation [Electronic resource] / Koha Community. – 2023. – Mode of access: https://koha-community.org/documentation/. – Date of access: 16.04.2025.

36. Aleph 500 System Administration Guide Version 23 [Electronic resource] / Ex Libris Group. – 2023. – Mode of access: https://knowledge.exlibrisgroup.com/Aleph. – Date of access: 17.04.2025.

37. RFC 6585: Additional HTTP Status Codes [Electronic resource] / M. Nottingham, R. Fielding // Internet Engineering Task Force (IETF). – 2012. – Mode of access: https://www.rfc-editor.org/rfc/rfc6585. – Date of access: 18.04.2025.

38. RFC 6797: HTTP Strict Transport Security (HSTS) [Electronic resource] / J. Hodges, C. Jackson, A. Barth // Internet Engineering Task Force (IETF). – 2012. – Mode of access: https://www.rfc-editor.org/rfc/rfc6797. – Date of access: 18.04.2025.

39. Federal Information Processing Standard 140-3: Security Requirements for Cryptographic Modules [Electronic resource] / National Institute of Standards and Technology (NIST). – Gaithersburg, MD : NIST, 2019. – Mode of access: https://csrc.nist.gov/publications/detail/fips/140/3/final. – Date of access: 20.04.2025.

40. Коршун, Н. В., Бондарчук, А. П., Складанний, П. М., Соколов, В. Ю., & Крижанівська, Т. М. (2026). Моделювання та реалізація атак соціальної інженерії. Телекомунікаційні та інформаційні технології, (1), 130-139.

41. National Vulnerability Database (NVD) [Electronic resource] / National Institute of Standards and Technology (NIST). – 2024. – Mode of access: https://nvd.nist.gov/. – Date of access: 20.04.2025.

42. ISO/IEC 27001:2022. Information Security, Cybersecurity and Privacy Protection – Information Security Management Systems – Requirements [Electronic resource] / International Organization for Standardization. – Geneva : ISO, 2022. – Mode of access: https://www.iso.org/standard/27001. – Date of access: 05.03.2025.

# ДОДАТОК А

Таблиця А.1

Відповідність LibroFlow вимогам OWASP ASVS v4.0.3 (рівень 2)

| Категорія | Вимога | Реалізація | Статус |
|---|---|---|---|
| V2 | V2.1.1 Мінімальна довжина пароля 10 символів | MinimumLengthValidator(min_length=10) | ✓ |
| V2 | V2.1.5 Самостійне скидання через верифікований канал | PasswordResetView + email-токен | ✓ |
| V2 | V2.1.6 Зміна пароля вимагає поточного | PasswordChangeView з old_password | ✓ |
| V2 | V2.1.10 Перевірка через HIBP API | Не інтегровано | ✗ |
| V2 | V2.2.1 Блокування після N невдалих спроб | django-axes, FAILURE_LIMIT=5 | ✓ |
| V2 | V2.2.4 Захист від credential stuffing | django-axes + AxesStandaloneBackend | ✓ |
| V2 | V2.7.1 Підтримка TOTP як другого фактора | Не реалізовано | ✗ |
| V3 | V3.2.1 Новий session token після автентифікації | request.session.cycle_key() у LoginView | ✓ |
| V3 | V3.2.3 Серверне зберігання сесійного стану | Django database session backend | ✓ |
| V3 | V3.3.1 Тривалість сесії обмежена | SESSION_COOKIE_AGE = 3600 (1 год) | ✓ |
| V3 | V3.3.2 Абсолютний тайм-аут | SESSION_COOKIE_AGE=3600 + SESSION_SAVE_EVERY_REQUEST=False | ✓ |
| V4 | V4.1.1 Централізована перевірка прав доступу | PermissionRequiredMixin / @permission_required у представленнях | ✓ |
| V4 | V4.1.2 Відмова за замовчуванням | PermissionDenied при відсутній ролі | ✓ |
| V4 | V4.1.3 Принцип мінімальних привілеїв | Три ролі з ієрархічними правами | ✓ |

| | | | |
|---|---|---|---|
| V4 | V4.1.5 Журналювання порушень доступу | @permission_required(raise_exception=True) → HTTP 403 | ✓ |
| V4 | V4.2.1 Захист від IDOR | get_object_or_404 + перевірка дозволів Django | ✓ |
| V4 | V4.2.2 Захист від CSRF | {% csrf_token %} + middleware | ✓ |
| V4 | V4.3.1 Restrict management interfaces | Адмін-панель за роллю admin | ✓ |
| V4 | V4.3.2 Directory browsing вимкнено | WhiteNoise: автоматичний лістинг каталогів вимкнено (статика віддається лише за маніфестом) | ✓ |
| V6 | V6.2.1 Затверджені алгоритми шифрування | gzip-стиснення, TLS 1.3 | ✓ |
| V6 | V6.2.2 Випадковий IV/nonce | secrets.token_urlsafe(16) | ✓ |
| V6 | V6.3.1 CSPRNG для токенів | secrets module (CSPRandom) | ✓ |
| V6 | V6.4.1 Управління ключами через .env | SECRET_KEY, GPG passphrase лише в .env | ✓ |
| V7 | V7.1.1 Журналювання подій безпеки | AuditLog: 16 типів, 3 рівні | ✓ |
| V7 | V7.1.2 Без чутливих даних у журналах | Паролі, session_key не зберігаються | ✓ |
| V7 | V7.2.1 Незмінність журналу | Перевизначені save()/delete() | ✓ |
| V7 | V7.2.2 Захист журналу від ін'єкцій | JSONField без конкатенації (параметризований ORM-запит) | ✓ |
| V7 | V7.3.1 Часова мітка | auto_now_add=True + DDL DEFAULT NOW() | ✓ |
| V7 | V7.4.1 Сповіщення про критичні події | SOC dashboard агрегує SEVERITY_CRITICAL | ✓ |
| V9 | V9.1.1 TLS 1.2 або вище | Django: SECURE_SSL_REDIRECT і SECURE_HSTS_SECONDS=31536000 за SECURE_SSL=true | ✓ |
| V9 | V9.1.2 Актуальні TLS-сертифікати | Let's Encrypt + certbot --renew | ✓ |

| | | | |
|---|---|---|---|
| V9 | V9.1.3 Заборона слабких шифр-сьютів | Явний allowlist хостів у ALLOWED_HOSTS (змінна оточення) | ✓ |
| V9 | V9.2.1 HSTS із includeSubDomains | max-age=31536000; includeSubDomains; preload | ✓ |