Міністерство освіти і науки України
Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра захисту інформації

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА
на тему:
«МЕТОД ТА ЗАСІБ АНАЛІЗУ БЕЗПЕКИ ВЕБСАЙТІВ»

Виконав: студент 2 курсу групи 1БС-24м
спеціальності 125 Кібербезпека та захист інформації
_____Іван КРАВЧУК

Керівник: к. т. н., доц., доцент каф. ЗІ
_____ Юрій БАРИШЕВ
«19» грудня 2025 р.

Опонент: к. т. н., доц., доцент каф. ПЗ
_____ Денис КАТЄЛЬНІКОВ
«19» грудня 2025 р.

Допущено до захисту
В. о. завідувача кафедри ЗІ
д. т. н., проф.
_____ Володимир ЛУЖЕЦЬКИЙ
«19» грудня 2025 р.

Вінниця ВНТУ – 2025 року

Міністерство освіти і науки України

Вінницький національний технічний університет

Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра захисту інформації
Рівень вищої освіти II (магістерський)
Галузь знань – 12 Інформаційні технології
Спеціальність – 125 Кібербезпека та захист інформації
Освітньо-професійна програма – Безпека інформаційних і комунікаційних систем

ЗАТВЕРДЖУЮ
В. о. завідувача кафедри ЗІ
д. т. н., проф.
Володимир ЛУЖЕЦЬКИЙ
«27» 09 2025 року

З А В Д А Н Н Я
НА МАГІСТЕРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

Кравчуку Івану Юрійовичу

1. Тема роботи: «Метод та засіб аналізу безпеки вебсайтів», керівник роботи: Баришев Юрій Володимирович, к.т.н., доцент кафедри ЗІ, затверджені наказом ректора ВНТУ від 24 вересня 2025 року № 313.
2. Строк подання студентом роботи 19 грудня 2025 р.
3. Вихідні дані до роботи:
   - Мова програмування Rust;
   - Об'єкт тестування: frontend, backend.
   - Тип бази даних документ-орієнтована.
4. Зміст текстової частини: Висновки. 1 Аналіз методів дослідження безпеки вебсайтів. 2 Метод аналізу безпеки вебсайтів. 3 Засіб аналізу безпеки вебсайтів. 4 Експериментальні дослідження та тестування. 5. Економічна частина. Список використаних джерел. Додатки.
5. Перелік ілюстративного матеріалу: аналіз засобів тестування на проникнення, аналіз сканерів вразливостей, математичний опис, метод аналізу безпеки вебсайтів, архітектура засобу аналізу безпеки вебсайтів, модель бази даних вразливостей, узагальнений алгоритм роботи засобу, алгоритм пасивної розвідки, алгоритм статичного аналізу, алгоритм динамічного аналізу, формати звітів, результати модульного тестування, результати роботи засобу, показники економічної ефективності.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		Завдання видав	Завдання прий
1	Ю. БАРИШЕВ, к.т.н., доц., доц. каф. ЗІ	25.09.25	24.10
2	Ю. БАРИШЕВ, к.т.н., доц., доц. каф. ЗІ	25.09.25	31.10
3	Ю. БАРИШЕВ, к.т.н., доц., доц. каф. ЗІ	25.09.25	14.11
4	Ю. БАРИШЕВ, к.т.н., доц., доц. каф. ЗІ	25.0925	28.11
5	О. ЛЕСЬКО, к.е.н., доц. зав. каф. ЕПВМ	25.09.25	2.12.25

7. Дата видачі завдання 24.09.25р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів магістерської кваліфікаційної роботи	Строк виконання етапів роботи	Приміт
1	Аналіз завдання. Вступ	24.09.2025 – 26.09.2025	
2	Аналіз інформаційних джерел за напрямком магістерської кваліфікаційної роботи	27.09.2025 – 07.10.2025	
3	Науково-технічне обґрунтування	11.10.2025 – 22.10.2025	
4	Аналіз методик та методів тестування безпеки вебсайтів	23.10.2025 – 26.10.2025	
5	Аналіз та формування вимог до засобу тестування	27.10.2025 – 02.11.2025	
6	Розробка методу тестування безпеки вебсайтів	03.11.2025 – 10.11.2025	
7	Застосування методу тестування безпеки вебзастосунків	10.11.2025 – 17.11.2025	
8	Розробка розділу економічної частини	18.11.2025 – 22.11.2025	
9	Оформлення пояснювальної записки	23.11.2025 – 29.11.2025	
10	Попередній захист та доопрацювання МКР	29.11.2025 – 11.12.2025	
11	Перевірка на наявність текстових запозичень.	12.12.2025 – 15.12.2025	
12	Представлення МКР до захисту, рецензування	16.12.2025 – 19.12.2025	
13	Захист МКР	19.12.2025 – 23.12.2025	

Студент _____ Іван КРАВ

Керівник роботи _____ Юрій БАРИШ

АНОТАЦІЯ

УДК 004.056

Кравчук І. Метод та засіб аналізу безпеки вебсайтів. Магістерська кваліфікаційна робота зі спеціальності 125 – Кібербезпека та захист інформації, освітня програма – Безпека інформаційних і комунікаційних систем. Вінниця: ВНТУ, 2025. 88 с.

Укр. мовою. Бібліогр.: 75 назв; рис.: 34; табл.: 25.

Магістерська кваліфікаційна робота присвячена розробці методу та засобу аналізу безпеки вебсайтів. Здійснено аналіз нормативно-правової бази тестування вебсайтів та методів тестування на проникнення в контексті їх використання для виявлення вразливостей. Здійснено математичний опис, побудовано алгоритми роботи та розроблено метод аналізу безпеки вебсайтів, який на відміну від відомих поєднує застосування різнорідних баз даних вразливостей з активними методами аналізу. Розроблено програмний засіб та проведено його тестування згідно з вимогами безпеки. Оцінено витрати та визначено економічну доцільність реалізації.

Ілюстративна частина складається з 14 плакатів з демонстрацією результатів проведеного тестування.

Ключові слова: аналіз безпеки, вебсайти, вразливості, статичний аналіз, динамічний аналіз, CVE, OWASP, сканер, кібербезпека.

ABSTRACT

UDC 004.056

Kravchuk I. Method and means of website security analysis. Master's thesis in the field of 125 – Cybersecurity and Information Protection, educational program – Security of Information and Communication Systems.Vinnytsia: VNTU, 2025. 88 p.

In Ukrainian language. Bibliographer: 75 titles; fig.: 34; tabl.: 25.

The master’s thesis is dedicated to the development of a method and tool for website security analysis. The regulatory and legal framework for website testing and penetration testing methods was analyzed in the context of their use for vulnerability detection. A mathematical description was provided, operational algorithms were constructed, and a website security analysis method was developed, which, unlike existing approaches, combines the use of heterogeneous vulnerability databases with active analysis techniques. A software tool was implemented and tested in accordance with security requirements. The costs were evaluated, and the economic feasibility of implementation was determined.

The illustrative section consists of 14 posters demonstrating the results of the conducted testing.

Keywords: security analysis, websites, vulnerabilities, static analysis, dynamic analysis, CVE, OWASP, scanner, cybersecurity.

ЗМІСТ

ВСТУП

У сучасному цифровому середовищі вебсайти є ключовим елементом інформаційної інфраструктури організацій, забезпечуючи обмін даними, надання послуг і взаємодію з користувачами. Зростання кількості вебсайтів та обсягу конфіденційної інформації, що передається через них, зумовлює підвищення ризиків кіберзагроз. Атаки на вебресурси можуть призвести до витоку даних, несанкціонованого доступу чи порушення доступності сервісів, що створює серйозні наслідки для бізнесу та державних структур.

Проблема забезпечення безпеки вебсайтів потребує комплексного підходу, який охоплює аналіз як клієнтської так і серверної частини вебсайтів. Традиційні засоби тестування на проникнення здебільшого орієнтовані на ручне тестування. Тому актуальним є розроблення методу й програмного засобу, які автоматизують процес аналізу.

Об’єктом дослідження є процес аналізу безпеки вебсайтів. Предметом дослідження є методи й засоби виявлення вразливостей вебсайтів.

Наукова новизна роботи полягає у тому, що удосконалено метод аналізу безпеки вебсайтів, який на відміну від відомих поєднує застосування різнорідних баз даних вразливостей з активними методами аналізу, що дозволяє збільшити кількість виявлених вразливостей.

Практична цінність роботи полягає у розробці програмного засобу аналізу безпеки вебсайтів.

Метою роботи є збільшення рівня безпеки вебсайтів за рахунок виявлення вразливостей.

Для досягнення поставленої мети необхідно виконати такі завдання:

– проаналізувати сучасні методи та інструменти сканування безпеки;

– розглянути нормативно-правову базу, що регламентує тестування безпеки;

– розробити математичний опис безпеки вебсайту;

– спроектувати архітектуру бази даних для зберігання інформації про вразливості;

– розробити та реалізувати метод аналізу вебсайтів;

– розробити формат звітування про результати сканування;

– виконати експериментальні дослідження;

– визначити показники економічної доцільності розробки.

Результати, отримані в роботі апробовувались на 2 конференціях:

– LIV Всеукраїнська науково-технічна конференція факультету інформаційних технологій та комп`ютерної інженерії, Вінниця, 2025 [1];

– Міжнародна конференція SMICS-2025 «Безпека сучасних інформаційно-комунікаційних систем», Львів, 2025 [2].

АНАЛІЗ МЕТОДІВ ДОСЛІДЖЕННЯ БЕЗПЕКИ ВЕБСАЙТІВ

1.1 Аналіз нормативно-правової бази

Забезпечення безпеки вебсайтів є критично важливою частиною кіберзахисту сучасних інформаційних систем, оскільки від роботи вебзастосунків залежить збереження конфіденційної інформації, стабільність бізнес-процесів та репутація організацій.

Проведення тестування та сканування вебсайтів є діяльністю, що потребує особливої уваги до правового аспекту. Будь-які дії, спрямовані на виявлення вразливостей або втручання в роботу вебзастосунку, можуть призвести до зміни його стану або порушення функціонування. Тому першочерговою умовою легальності таких дій є наявність письмового дозволу від власника ресурсу або уповноваженої особи. В Україні будь-яке тестування без письмового дозволу власника ресурсу вважається несанкціонованим втручанням і підпадає під кримінальну відповідальність [3].

Нормативно-правове регулювання у сфері кібербезпеки базується на законах України, які визначають межі допустимих дій та відповідальність за порушення. Ключові положення містяться в Кримінальному кодексі України, де прямо передбачено покарання за несанкціонований доступ, зміну або знищення інформації [3]:

– Стаття 361 - несанкціоноване втручання в роботу комп’ютерних систем чи мереж, що призвело до спотворення, знищення або блокування інформації;

– Стаття 361-1 - створення, використання або розповсюдження шкідливих програмних засобів;

– ст. 361-2 - несанкціоноване розповсюдження інформації з обмеженим доступом;

– Стаття 362 - несанкціоновані дії з інформацією, яка обробляється в автоматизованих системах;

– Стаття 363 - порушення правил експлуатації комп’ютерних систем, що спричинило несанкціоновану зміну, втрату або витік інформації.

Будь-яке тестування вебсайтів можливе лише за письмовим дозволом власника ресурсу, що може бути оформлено через договір на проведення аудиту безпеки або в межах офіційних програм Bug Bounty, які визначають дозволені дії, порядок повідомлення про вразливості та умови взаємодії між пентестером і власником ресурсу.

Bug Bounty програми забезпечують легальне тестування вебзастосунків, дозволяючи пентнестерам перевіряти захищеність ресурсів без порушення закону, а організаціям отримувати інформацію про потенційні ризики до того, як ними можуть скористатися зловмисники [4].

Для регламентації процесу офіційного повідомлення про вразливості використовується стандарт /.well-known/security.txt. Він визначає, як пентестер може звернутися до служби безпеки організації, повідомити про знайдену вразливість і отримати зворотний зв’язок [5]. Стандарт виник через проблему, коли незалежні пентестери виявляли серйозні ризики, але не мали чітких каналів для їх повідомлення. На рис 1.1 наведено типовий вміст файлу security.txt.

```
# Our security address
Contact: mailto:security@example.com

# Our OpenPGP key
Encryption: https://example.com/pgp-key.txt

# Our security policy
olicy: https://example.com/security-policy.html

# Our security acknowledgments page
Acknowledgments: https://example.com/hall-of-fame.html
```

Рисунок 1.1 – Типовий вміст файлу security.txt

Такий файл допомагає пентестерам швидко знайти контактні дані служби безпеки та зрозуміти правила взаємодії з організацією. Нижче наведено пояснення основних полів:

– Contact — електронна адреса або інший спосіб для повідомлення про вразливості;

– Policy — посилання на політику відповідального розкриття, яка визначає умови, порядок і межі тестування;

– Encryption — публічний ключ для шифрування повідомлень, що підвищує конфіденційність комунікації;

– Acknowledgments — сторінка з винагородами пентестерам, що стимулює участь у програмі та офіційно визнає їх внесок;

– Preferred-Languages — переважні мови для комунікації, що допомагає організувати ефективний обмін інформацією.

Розміщення файлу security.txt на вебсайті свідчить про відкритість власника ресурсу до повідомлень про вразливості та готовність дозволяти тестування в межах встановлених правил. Це створює прозорий канал зв’язку між пентестером і власником, знижує ризики юридичної відповідальності та підвищує ефективність усунення виявлених вразливостей.

Використання security.txt допомагає встановити чіткий і зрозумілий порядок взаємодії між тестувальниками безпеки та власниками вебресурсів. У ньому можна визначити типи вразливостей, що дозволені до тестування, обмеження щодо перевірки продуктивних систем і строки подання звітів. Такий підхід забезпечує прозорість процесу дослідження та сприяє дотриманню законодавства України, яке встановлює відповідальність за несанкціоноване втручання у вебсайти.

1.2 Аналіз вебсайтів

У сучасному цифровому середовищі вебсайти забезпечують інтеграцію користувачів із сервісами, базами даних та бізнес-логікою, формуючи основу більшості онлайн-рішень. Вони складаються з клієнтської частини, що відповідає за інтерфейс та взаємодію з користувачем і серверної частини, яка обробляє запити, зберігає дані та виконує бізнес-логіку. Клієнтська частина включає HTML, CSS та JavaScript, що формують візуальний та інтерактивний інтерфейс, тоді як серверна забезпечує аутентифікацію, авторизацію, роботу з базою даних і обробку API-запитів [6].

Взаємодія між клієнтською і серверною частиною відбувається через запити та відповіді, де клієнт надсилає інформацію, а сервер обробляє її відповідно до логіки додатку. На рисунку 1.2 наведено приклад роботи вебсайтів.

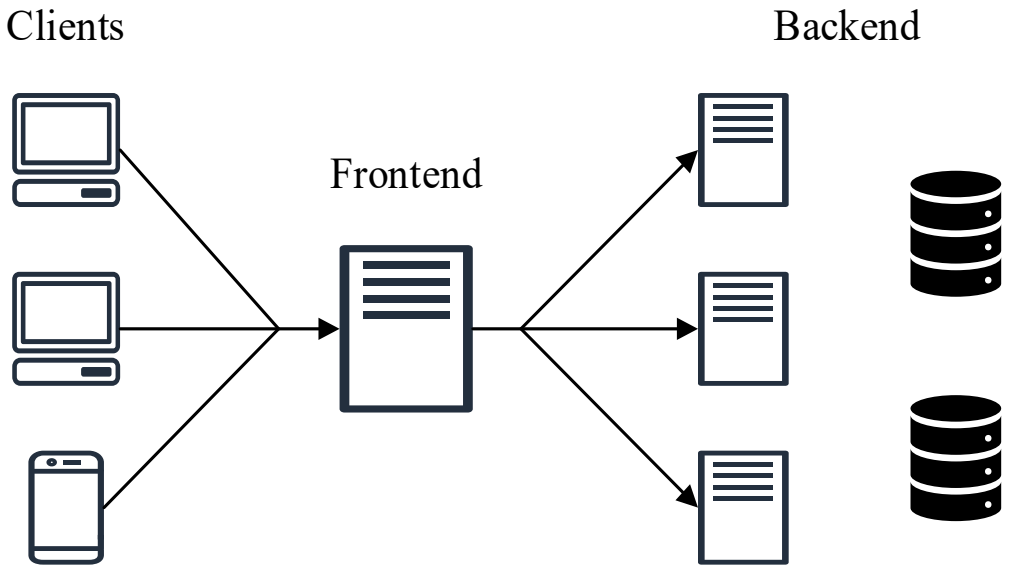


Рисунок 1.2 – Схема взаємодії клієнтської та серверної частин вебсайту

У процесі взаємодії клієнта та сервера використовуються HTTP методи, які визначають тип операції над ресурсами. Знання дозволених методів дозволяє зрозуміти логіку роботи вебсайту, а також оцінити потенційні вектори атак [7,8]. В таблиці 1.1 наведено HTTP-методи.

Таблиця 1.1 – HTTP методи

Метод	**Опис**
GET	Використовується для отримання інформації з сервера
POST	Застосовується для надсилання даних на сервер
PUT	Дозволяє оновлювати або створювати ресурси
DELETE	Використовується для видалення ресурсів
PATCH	Застосовується для часткового оновлення ресурсів
OPTIONS	Використовується для перевірки доступних методів
HEAD	Отримує лише заголовки відповіді

З таблиці 1.1 видно, що кожен метод виконує окрему функцію у взаємодії між клієнтом і сервером. Розуміння дозволених методів допомагає визначити

потенційні точки впливу на вебсайт і вибрати оптимальні техніки тестування безпеки.

Також важливо звертати увагу на коди відповідей сервера, які сигналізують про результат виконання запиту. Наприклад, помилка 404 може вказувати на відсутність ресурсу, тоді як код 403 – на наявність прихованого ресурсу, доступ до якого обмежено [8]. В таблиці 1.2 наведено поширені HTTP-коди відповідей.

Таблиця 1.2 – Поширені HTTP-коди відповідей

Код	**Назва**	**Опис**
200	OK	Запит виконано успішно
301 / 302	Redirect	Відбулося перенаправлення на іншу сторінку
400	Bad Request	Сервер не зміг обробити запит
401	Unauthorized	Необхідна автентифікація
403	Forbidden	Доступ до ресурсу заборонено
404	Not Found	Сторінка не знайдена
405	Method Not Allowed	Метод запиту не підтримується сервером
500	Internal Server Error	Помилка в роботі сервера
503	Service Unavailable	Сервер тимчасово недоступний

Як видно з таблиці 1.2, коди відповіді допомагають визначити стан обробки запиту сервером. Аналіз таких кодів дозволяє виявити приховані ресурси, неправильні налаштування доступу або логічні помилки в роботі вебсайту.

Ще одним важливим аспектом аналізу є HTTP заголовки, які визначають політику безпеки, тип контенту, правила кешування та доступу між доменами. Одним із ключових механізмів є Cross-Origin Resource Sharing (CORS), який регулює взаємодію між різними джерелами [9, 10]. Неправильна конфігурація CORS може призвести до витоку даних або обходу обмежень доступу. В таблиці 1.3 наведено налаштування CORS.

Таблиця 1.3 – HTTP-заголовки безпеки

Заголовок	**Опис**
Access-Control-Allow-Origin	Визначає, які домени можуть отримувати доступ до ресурсу
Access-Control-Allow-Methods	Визначає дозволені HTTP методи
Access-Control-Allow-Headers	Визначає, які заголовки можуть бути використані
Access-Control-Allow-Credentials	Дозволяє передачу cookies або токенів
Access-Control-Max-Age	Визначає час кешування CORS-політики
Access-Control-Expose-Headers	Визначає, які заголовки доступні клієнту
Origin	Вказує домен, з якого здійснено запит

З таблиці 1.3 можна побачити, що налаштування заголовків безпеки безпосередньо впливають на захист вебзастосунка. Неправильна конфігурація може дозволити обхід політик доступу або витік конфіденційних даних.

Для управління індексацією сторінок пошуковими системами використовують robots.txt. Цей файл визначає, які сторінки можна індексувати, а які заборонено. Він може містити шляхи до адміністративних панелей, тестових директорій або інших конфіденційних ресурсів, що потенційно можуть бути вразливими [11]. На рис 1.3 наведено вміст файлу robots.txt.

```
# Забороняємо індексацію конфіденційних директорій
User-agent: *
Disallow: /admin/
Disallow: /test/
Disallow: /private/

# Дозволяємо доступ до загальних ресурсів
Allow: /public/

Sitemap: https://example.com/sitemap.xml
```

Рисунок 1.3 – Вміст файлу robots.txt

Файл sitemap.xml є доповненням до robots.txt, оскільки він містить структуру сайту та перелік усіх сторінок, доступних для індексації. Під час аналізу sitemap.xml можна знайти сторінки, що не відображаються у навігації,

проте залишаються активними [12]. Це дає змогу отримати повніше уявлення про логіку побудови сайту, а також виявити потенційні вразливі або приховані розділи.

Одним із ключових елементів клієнтської частини вебсайту є Document Object Model (DOM), який представляє структуру HTML-документа як дерево об'єктів, доступне для маніпуляції через JavaScript. DOM є інтерфейсом програмування додатків, що дозволяє динамічно змінювати вміст, структуру та стиль вебсторінки в браузері користувача без необхідності перезавантаження сторінки. Він формується браузером на основі розпарсеного HTML-коду і слугує основою для інтерактивних вебзастосунків [13]. На рисунку 1.4 представлено архітектуру DOM.

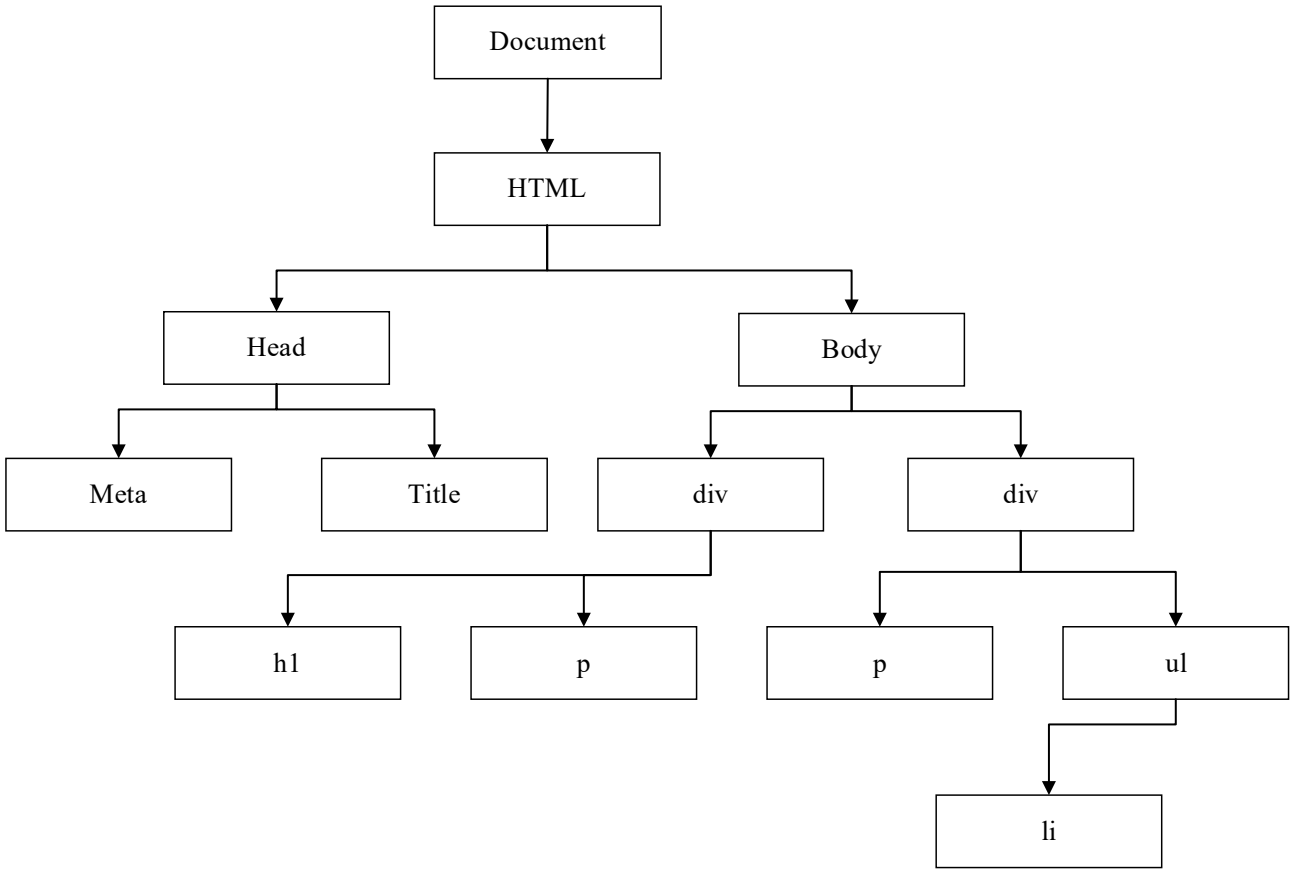


Рисунок 1.4 – Архітектура DOM

З рисунку 1.4 видно, що DOM складається з ієрархічної структури вузлів. Кожен вузол представляє певну частину документа і може мати батьківські, дочірні та сусідні вузли. Основні типи вузлів включають елементні вузли, що

представляють HTML-теги і містять атрибути. Ця структура дозволяє JavaScript взаємодіяти з DOM через методи, такі як getElementById(), appendChild() або addEventListener(), роблячи вебсторінки динамічними. Однак така гнучкість відкриває потенційні вектори для атак, особливо якщо дані від користувача не фільтруються належним чином.

Вектори атак на DOM переважно пов'язані з клієнтською стороною і спрямовані на маніпуляцію структурою документа в браузері жертви, дозволяючи зловмисникам виконувати шкідливий код, викрадати дані або обманювати користувача. Для оцінки потенційних загроз безпеки розглянуто основні типи атак, які можуть бути застосовані до DOM, та їхні наслідки для користувачів і системи [13]. Основні вектори атак на DOM наведені в таблиці 1.4.

Таблиця 1.4 – Вектори атак на DOM

Вектор атаки	**Опис**
Cross-Site Scripting (XSS)	Вставка шкідливого JavaScript-коду в DOM через невалідовані дані, такі як поля вводу чи URL-параметри, за допомогою цього можна викрасти cookies, сеансові токени, виконати довільний код або перенаправити користувача на шкідливі сайти.
Clickjacking	Обман користувача через приховані iframe або елементи в DOM; за допомогою цього можна здійснювати несанкціоновані кліки, запити або взаємодії від імені користувача.
DOM Clobbering	Маніпуляція іменами елементів або атрибутів у DOM для перезапису глобальних об’єктів JavaScript, за допомогою цього можна виконувати шкідливий код або змінювати поведінку скриптів.
Mutation-based Attacks	Використання динамічних змін у DOM для обходу захисних механізмів, за допомогою цього можна обійти політики безпеки та виконувати заборонені дії.
Event Handler Injection	Вставка шкідливих обробників подій у DOM через невалідовані дані, за допомогою цього можна виконувати код при взаємодії користувача, наприклад, при кліку або наведенні миші.
Prototype Pollution	Маніпуляція прототипами об’єктів у JavaScript для впливу на DOM, за допомогою цього можна виконувати довільний код.

Таблиця 1.4 демонструє основні вектори атак на DOM та потенційні наслідки. Особливу увагу слід приділити Cross-Site Scripting, оскільки це найпоширеніший вектор атак на DOM, який становить значну загрозу через свою універсальність і простоту реалізації. XSS дозволяє зловмиснику вставляти

шкідливий JavaScript-код у DOM через не перевірені дані, отримуючи доступ до cookies, токенів, особистих даних, або змінюючи вміст сторінки. Основні види XSS включають [14]:

– Reflected XSS – Шкідливий код передається через URL або форми, відображається сервером і виконується одразу після завантаження сторінки.

– Stored XSS – Код зберігається на сервері, наприклад, у коментарях чи профілях, і виконується для всіх користувачів сторінки, що забезпечує масовий вплив.

– DOM-based XSS – Атака відбувається на клієнтській стороні, коли JavaScript обробляє невалідовані дані з URL чи localStorage, вставляючи їх у DOM.

Для запобігання XSS-атакам слід ретельно перевіряти введені користувачем дані та екранувати їх перед відображенням у браузері.

Окрім атак на DOM, потрібно врахувати вектори атак на серверну частину, які спрямовані на сам сервер, базу даних або бізнес-логіку і можуть мати більш серйозні наслідки. Ці атаки часто експлуатують вразливості в обробці даних або конфігурації сервера, що дозволяє зловмисникам отримати несанкціонований доступ до ресурсів [15]. Вектори атак на серверну частину наведені в таблиці 1.5.

Таблиця 1.5 – Вектори атак на сервер

Вектор атаки	**Опис**
SQL Injection (SQLi)	Вставка шкідливого SQL-коду в запити до бази даних через невалідовані вхідні дані; за допомогою цього можна читати, модифікувати або видаляти дані з бази даних.
Cross-Site Request Forgery (CSRF)	Підроблені запити від імені користувача для виконання несанкціонованих дій; за допомогою цього можна змінювати дані, наприклад, паролі чи проводити транзакції.
Remote Code Execution (RCE)	Виконання довільного коду на сервері через вразливості, такі як небезпечні функції або завантаження файлів; за допомогою цього можна отримати повний контроль над системою.
Server-Side Request Forgery (SSRF)	Змушення сервера надсилати запити до внутрішніх або зовнішніх ресурсів; за допомогою цього можна витікати дані або атакувати інфраструктуру.
Directory Traversal	Доступ до файлів поза дозволеною директорією через маніпуляцію шляхами; за допомогою цього можна красти конфіденційні файли, наприклад, конфігураційні дані.

Продовження таблиці 1.5

Вектор атаки	Опис
XML External Entity (XXE)	Експлуатація XML-парсерів для читання локальних файлів або виконання DoS; за допомогою цього можна витікати дані або викликати відмову в обслуговуванні.
Insecure Direct Object References (IDOR)	Доступ до об’єктів через маніпуляцію ідентифікаторами в запитах; за допомогою цього можна несанкціоновано доступатися до даних інших користувачів.
File Inclusion Attacks	Включення шкідливих файлів у серверний код через вразливості в обробці шляхів; за допомогою цього можна виконувати код або витікати дані.

Для запобігання атак на серверну частину слід використовувати CSRF-токени, обмеження доступу до файлів, підготовлені запити для SQL і перевірку ідентифікаторів об’єктів.

Проведений аналіз клієнтської та серверної частин вебсайтів показує, що вразливості в HTTP-методах, заголовках, DOM і серверна логіка створюють ризики витоку даних, несанкціонованого доступу чи порушення доступності. Цей аналіз є основою для розробки методів захисту, які будуть розглянуті в наступних розділах роботи.

1.3 Порівняльний аналіз засобів тестування на проникнення

Тестування на проникнення є ключовим етапом оцінювання безпеки вебсайтів, спрямованим на виявлення вразливостей шляхом імітації дій зловмисників у реальних умовах. Ці засоби належать до категорії DAST, яка аналізує вебсайти через маніпуляцію HTTP-запитами, перевірку вхідних даних і симуляцію атак. Вони дозволяють тестувальникам знаходити як стандартні вразливості, так і складні логічні помилки які часто не може виявити статичний сканер.

Ефективність таких інструментів полягає в їхній здатності поєднувати автоматизацію для швидкого сканування з ручними методами для глибокого аналізу, що критично для сучасних вебзастосунків, побудованих на JavaScript-фреймворках, API чи мікросервісах. Вони дозволяють налаштувати процес

тестування для специфічних сценаріїв, таких як перевірка хмарних сервісів чи мобільних інтерфейсів, забезпечуючи гнучкість для різних типів проєктів.

Процес тестування охоплює кілька етапів: розвідку для збору інформації про ціль, сканування для ідентифікації слабких місць і експлуатацію для оцінки впливу вразливостей. Це дає змогу не лише виявляти проблеми і продемонструвати їхній потенційний вплив, що допомагає організаціям визначити пріоритети заходів безпеки. Завдяки адаптивності цих засобів, тестувальники можуть працювати з різними масштабами систем, від невеликих сайтів до великих платформ [16, 17].

Гнучкість інструментів дозволяє тестувальникам адаптувати підходи до конкретних потреб, наприклад для тестування аутентифікації, API чи клієнтських компонентів. Це робить їх незамінними в умовах зростання кіберзагроз, таких як нові вектори атак на серверну чи клієнтську сторони, забезпечуючи комплексний захист вебсайтів. У таблиці 1.6 наведено короткі описи засобів тестування на проникнення.

Таблиця 1.6 – Засоби тестування на проникнення

Засіб	**Опис**
Acunetix	Автоматизований сканер вебзастосунків, який швидко виявляє вразливості, забезпечуючи детальні звіти з рекомендаціями щодо виправлення проблем. Підтримує автоматизацію сканування для великих проєктів і має низький рівень помилкових результатів [18].
Aircrack-ng	Інструмент для аналізу безпеки Wi-Fi мереж, що дозволяє тестувати протоколи WEP/WPA/WPA2 шляхом злому ключів, перехоплення пакетів і моніторингу трафіку [19]. Він ефективний для перевірки бездротових мереж, але потребує спеціального обладнання.
Astra Pentest	Платформа, що поєднує автоматизоване та ручне тестування вебзастосунків, надаючи детальні звіти для відповідності стандартам, таким як PCI DSS. Підходить для компаній, які потребують комплексного аналізу безпеки з підтримкою командної роботи [20].
Burp Suite	Потужний інструмент для перехоплення HTTP-трафіку, аналізу вебзастосунків і гнучкого тестування з підтримкою плагінів для розширення функціоналу. Він підтримує як автоматизоване, так і ручне тестування, що робить його популярним серед професіоналів безпеки [21].
DirBuster	Інструмент для пошуку прихованих директорій і файлів на вебсерверах через brute-force із підтримкою словників і графічного інтерфейсу [22]. Він ефективний для розвідки, але застарілий порівняно з сучасними альтернативами.

Продовження таблиці 1.6

Засіб	Опис
Ffuf	Швидкий fuzzer для тестування вебзастосунків, що дозволяє гнучко налаштовувати словники для пошуку вразливостей у параметрах, шляхах чи API [23].
Gobuster	Інструмент для швидкого пошуку директорій і файлів на серверах через brute-force, із підтримкою словників і відкритим кодом [24]. Він оптимізує швидкість розвідки, але обмежений пошуком шляхів.
Hashcat	Високопродуктивний інструмент для злому хешів паролів, підтримує широкий спектр алгоритмів. Він ефективний для офлайн-атак, але потребує потужного обладнання [25].
Hydra	Інструмент для brute-force атак на паролі через протоколи, такі як SSH, FTP, HTTP, що забезпечує високу швидкість тестування [26]. Підходить для перевірки слабких паролів, але може викликати блокування IP.
Intruder	Хмарний інструмент для автоматизованого тестування вебзастосунків і fuzzing, з акцентом на простоту використання та швидкість [27]. Підходить для швидкого аналізу, але має обмежені налаштування.
John the Ripper	Інструмент із відкритим кодом для злому паролів на CPU, підтримує різні формати хешів і словникові атаки. Ефективний для простих сценаріїв, але повільніший за GPU-альтернативи [28].
Kismet	Інструмент для моніторингу Wi-Fi мереж, що дозволяє виявляти пристрої, аналізувати трафік і працювати з дронами. Корисний для розвідки бездротових мереж, але потребує спеціального обладнання [29].
Metasploit	Фреймворк для створення, тестування та виконання експлойтів із великою базою модулів для різних типів атак [30]. Підтримує командну роботу і є стандартом для тестування на проникнення.
Nikto	Сканер для швидкого аналізу конфігурацій вебсерверів, що виявляє базові вразливості, такі як застарілі версії чи неправильні налаштування [31]. Простий у використанні, але обмежений для динамічних додатків.
Nmap	Інструмент для розвідки мереж і сканування портів із підтримкою скриптів для аналізу служб і вразливостей. Базовим інструментом для мережевої розвідки [32].
Nuclei	Сканер на основі шаблонів для швидкого виявлення вразливостей, підтримує YAML-конфігурації та відкритий код. Він дозволяє налаштувати тести, але потребує ручного налаштування шаблонів [33].
OWASP ZAP	Безкоштовний інструмент із відкритим кодом для активного та пасивного сканування вебзастосунків, підтримуваний спільнотою [34]. Підходить для початківців, але може генерувати помилкові результати.
RustScan	Швидкий сканер портів із відкритим кодом, що інтегрується з Nmap для ефективної розвідки мереж. Потребує додаткових інструментів для аналізу [35].
sqlmap	Інструмент для автоматизованого тестування SQL-ін'єкцій, що підтримує різні системи керування базами даних. Ефективний для спеціалізованих атак, але обмежений SQL-сценаріями [36].
Wireshark	Аналізатор мережевого трафіку, що забезпечує детальний моніторинг і фільтрацію пакетів для виявлення аномалій. Він корисний для пасивного аналізу, але потребує ручної перевірки даних [37].

Як видно з таблиці 1.6, кожен інструмент має унікальну функціональність, що дозволяє тестувальникам обирати засоби залежно від цілей тестування, таких

як аналіз вебзастосунків, мережева розвідка чи тестування паролів. Це забезпечує гнучкість і точність у виявленні вразливостей.

Для використання засобів тестування на проникнення часто застосовуються спеціалізовані дистрибутиви, які надають комплексне середовище з попередньо встановленими інструментами. Ці дистрибутиви адаптовані до процесу тестування, забезпечуючи доступ до широкого спектра засобів для різних сценаріїв безпеки [38]. У таблиці 1.7 наведено описи таких дистрибутивів.

Таблиця 1.7 – Описи дистрибутивів операційних систем для тестування на проникнення

Дистрибутив	**Опис**
Kali Linux	Найпопулярніший дистрибутив для тестування на проникнення, що включає широкий набір інструментів. Орієнтований на початківців і професіоналів, підтримує регулярні оновлення, але може бути ресурсомістким для слабких систем [39].
Parrot Security	Багатофункціональний дистрибутив на базі Debian, що підтримує тестування безпеки, аналіз мереж. Пропонує хмарні та легкі версії для різних сценаріїв, але потребує налаштування для специфічних завдань [40].
BlackArch Linux	Linux дистрибутив на базі Arch Linux, що інструментів для тестування безпеки, включаючи засоби для мережевої розвідки, аналізу вразливостей і експлуатації [41]. Підходить для досвідчених користувачів, які цінують гнучкість і можливість налаштування, але вимагає знань Arch Linux для ефективного використання.
Kali Purple	Дистрибутив Kali Linux, адаптований для "фіолетової команди" (blue-red team), що поєднує інструменти для захисту (blue team) і тестування на проникнення (red team). Включає засоби для моніторингу, аналізу і симуляції атак, підходящий для тренувань і гібридних сценаріїв безпеки, але вимагає базових знань Kali [42].
BackBox	Linux-дистрибутив на базі Ubuntu, орієнтований на аудит безпеки та тестування на проникнення з набором інструментів [43]. Легкий і простий у використанні, що робить його зручним для початківників, але має меншу кількість інструментів порівняно з Kali Linux.
DEFT Linux	Дистрибутив, орієнтований на цифрову криміналістику та тестування безпеки, із підтримкою аналізу мереж і відновлення даних. Він включає інструменти, як-от Wireshark і Autopsy, не зосереджений на активному тестуванні на проникнення порівняно з іншими дистрибутивами [44].
Pentoo	Дистрибутив на базі Gentoo, що пропонує інструменти для тестування безпеки, із акцентом на продуктивність. Він підходить для досвідчених користувачів, готових витратити час на налаштування [45].

З таблиці 1.7 видно, що спеціалізовані дистрибутиви забезпечують комплексне середовище для тестування на проникнення, надаючи попередньо встановлені інструменти для різних сценаріїв, від мережевої розвідки до аналізу вразливостей. Їхня адаптивність дозволяє ефективно використовувати їх у професійних аудитах безпеки, забезпечуючи зручність і високу продуктивність.

Проведений аналіз засобів і дистрибутивів для тестування на проникнення підкреслює важливість вибору інструментів і платформ, що відповідають конкретним потребам проєкту.

1.4 Порівняльний аналіз засобів виявлення вразливостей

Засоби виявлення вразливостей є основою планового моніторингу безпеки інфраструктури, мереж і серверів. Вони належать до категорії мережевих сканерів, призначених для регулярного аудиту великих систем, з акцентом на виявлення конфігураційних помилок, застарілих компонентів і системних загроз.

Ці інструменти цінні завдяки автоматизації перевірок, що дозволяє проводити регулярні сканування з мінімальними зусиллями, і підтримці стандартів таких як PCI DSS [46] чи ISO 27001 [47]. Вони ефективно виявляють системні проблеми, як-от незахищені порти чи відсутність патчів, забезпечуючи базовий рівень безпеки.

Обмеження сканерів включають високе навантаження на ресурси, повільність у великих мережах і схильність до помилкових результатів, що вимагає ручної верифікації. Вони менш придатні для вебзастосунків, але доповнюють DAST-інструменти для комплексного захисту.

Сканери інтегруються з SIEM-системами, надають детальні звіти з рекомендаціями та підтримують регулярне планування перевірок. Це забезпечує безперервний контроль безпеки в динамічних середовищах із новими активами чи змінами конфігурацій. У таблиці 1.8 наведено опис засобів виявлення вразливостей.

Таблиця 1.8 – Описи засобів виявлення вразливостей

Засіб	**Опис**
Greenbone	Безкоштовна платформа з вебінтерфейсом для сканування вразливостей у мережах, що підтримує налаштування і аналіз середніх за розміром проєктів. Інтегрується з базами вразливостей і підходить для організацій із обмеженим бюджетом, забезпечуючи регулярний аудит безпеки [48].
Nessus	Потужний сканер вразливостей із великою базою даних, що підтримує масштабованість, інтеграцію з SIEM і детальні звіти для великих мережевих інфраструктур [49]. Ефективний для планового аудиту і відповідності стандартам безпеки, таким як PCI DSS.
Nexpose	Інструмент для сканування мереж із визначення пріоритетності ризиків, інтеграцією з InsightVM і підтримкою великих мережевих інфраструктур. Дозволяє ефективно керувати вразливостями в складних середовищах, але потребує значних ресурсів [50].
OpenSCAP	Безкоштовний інструмент для перевірки конфігурацій операційних систем, що підтримує стандарти NIST/DISA. Ефективний для аналізу системного рівня, але обмежений для мережевих перевірок і потребує ручного налаштування [51].
OpenVAS	Безкоштовний сканер із відкритим кодом, що підтримує налаштування та регулярне оновлення бази вразливостей [52]. Підходить для базового аудиту, але менш точний порівняно з комерційними рішеннями.
Qualys VMDR	Хмарна платформа для управління вразливостями, що підтримує AWS/Azure, визначення пріоритетності ризиків і детальні звіти для великих мереж. Оптимізує автоматизований аудит, але залежить від хмарної інфраструктури [53].
QualysGuard	Хмарний інструмент для автоматизованого сканування великих мереж, з підтримкою відповідності стандартам і генерацією детальних звітів. Ефективний для великих організацій, але має обмежену гнучкість для специфічних сценаріїв [54].
Rapid7 InsightVM	Інструмент для управління вразливостями, що пропонує визначення пріоритетності ризиків, інтеграцію з SIEM і asset management для великих мереж [55]. Підходить для складних інфраструктур, але потребує ретельного налаштування.
Retina Network Scanner	Сканер для середніх мереж із підтримкою відповідності стандартам і автоматизованими звітами, орієнтований на виявлення системних вразливостей [56].
Tenable.io	Хмарна платформа для сканування вразливостей із підтримкою AWS/Azure, автоматизацією і відповідністю стандартам. Оптимальна для хмарних середовищ, але залежить від постачальника [57].
Tripwire IP360	Інструмент для enterprise-систем, що забезпечує управління активами, пріоритизацію ризиків і аудит великих мереж. Підходить для складних інфраструктур, але складний для малого бізнесу [58].
Wiz	Хмарний сканер для швидкого аналізу безпеки в AWS/Azure/GCP, з акцентом на відповідність стандартам і захист хмарних середовищ [59].

Як видно з таблиці 1.8, засоби забезпечують плановий аудит безпеки, охоплюючи інфраструктуру та мережі. Поєднання їх із DAST-інструментами

дозволяє комплексно захищати системи, виявляючи та усуваючи вразливості, адаптуючись до змін у мережевому середовищі.

1.5 Аналіз баз даних відомих вразливостей

Аналіз баз даних відомих вразливостей є важливою частиною тестування безпеки вебсайтів, оскільки дозволяє систематизувати знання про поширені загрози, оцінити їхній вплив та розробити ефективні методи захисту. Ці бази даних накопичують інформацію про виявлені вразливості, експлойти та рекомендації щодо їх усунення, що допомагає фахівцям з кібербезпеки проводити активний моніторинг і аудит систем.

Одним із ключових ресурсів для аналізу вразливостей вебзастосунків є OWASP Top 10 це список найкритичніших ризиків безпеки, який складений компанією Open Web Application Security Project [60]. Цей список базується на даних з реальних інцидентів і регулярно оновлюється для відображення актуальних тенденцій у кіберзагрозах. OWASP Top 10 фокусується на вебзастосунках, охоплюючи як клієнтську так і серверну частину.

Версія OWASP Top 10 2021 року представляє еволюцію попередніх списків, враховуючи нові вектори атак, такі як проблеми з цілісністю даних та серверними запитами. Кожна вразливість у списку оцінюється за ступенем поширеності, виявлення та впливу. Аналіз OWASP Top 10 допомагає зрозуміти, що більшість атак експлуатують базові помилки, такі як відсутність валідації вхідних даних чи неправильна конфігурація, які можна запобігти за допомогою стандартних практик безпеки. У таблиці 1.9 наведено опис вразливостей визначених стандартом OWASP Top 10 [61].

Таблиця 1.9 – OWASP Top 10 2021

№	Вразливість	Опис
A01:2021	Broken Access Control	Недоліки контролю доступу, що дозволяють обхід обмежень
A02:2021	Cryptographic Failures	Використання слабких або відсутніх механізмів шифрування

Продовження таблиці 1.9

№	Вразливість	Опис
A03:2021	Injection	Вставка шкідливого коду в запити (SQL, XXE, OS тощо)
A04:2021	Insecure Design	Недоліки у проєктуванні архітектури системи
A05:2021	Security Misconfiguration	Помилки налаштування серверів і сервісів
A06:2021	Vulnerable and Outdated Components	Використання застарілих або вразливих бібліотек
A07:2021	Identification and Authentication Failures	Проблеми з автентифікацією користувачів
A08:2021	Software and Data Integrity Failures	Маніпуляції з кодом або даними без перевірки цілісності
A09:2021	Security Logging and Monitoring Failures	Відсутність логування або моніторингу безпеки
A10:2021	Server-Side Request Forgery (SSRF)	Відправлення запитів до внутрішніх ресурсів сервера

З таблиці 1.9 видно, що OWASP Top 10 охоплює широкий спектр вразливостей, що робить його універсальним інструментом для оцінки безпеки вебсайтів. Наприклад Broken Access Control є найпоширенішою загрозою, оскільки дозволяє несанкціонований доступ до ресурсів, тоді як Injection залишається класичним вектором для компрометації баз даних. Використання OWASP Top 10 у поєднанні з інструментами тестування описаними в попередніх підрозділах дозволяє проводити комплексний аналіз до конкретних вебзастосунків.

Для створення сканера вразливостей важливо використовувати бази даних, які містять інформацію про відомі CVE та інші типи загроз. Бази даних регулярно оновлюються і зазвичай надають API для автоматизованої інтеграції, що дає змогу підтримувати актуальність інформації в інструментах тестування.

Використання таких баз у процесі сканування та аудиту автоматизує виявлення вразливостей, порівняння версій програмного забезпечення з відомими вразливими версіями. Крім того, комбінація кількох джерел забезпечує ширше охоплення вразливостей і дозволяє отримати додаткові дані про вразливості. В таблиці 1.10 наведено основні бази даних, які доцільно використовувати для аналізу вебсайтів [62-67].

Таблиця 1.10 – Бази даних відомих вразливостей

Назва бази	Короткий опис	Формат даних	Недоліки
CVE (MITRE)	Реєстр вразливостей з унікальними CVE-ID	CSV / JSON	Надає переважно базові описи і не містить даних для експлуатації вразливостей.
NVD (NIST)	Розширює CVE метаданими та CVSS для пріоритизації	JSON / REST API	Має ліміти запитів і іноді затримки в появі нових даних
OSV	Фіди для open-source пакетів і інтеграції зі SBOM	REST API JSON	Орієнтована лише на відкриті бібліотеки і вимагає нормалізації версій
Exploit-DB	Репозиторій PoC і експлойтів для оцінки експлуатабельності	CSV / GitHub mirror / вебпошук	Не надає повноцінного REST API і вимагає парсингу дампів або HTML
Vulners	Агрегатор CVE, advisories і експлойтів з різних джерел	REST API JSON	Безкоштовний доступ має суворі ліміти і деякі дані доступні лише в платних планах
VulnDB	Платна база з розширеними описами і підтримкою	Платний REST API JSON	Доступ лише за підпискою з контрактними лімітами

З таблиці 1.10 видно, що різні бази даних містять унікальну інформацію, яка варіюється від базових описів вразливостей до практичних експлойтів і рекомендацій щодо виправлення. Наприклад, CVE та NVD забезпечують стандартизовані ідентифікатори та загальний опис і рекомендації, тоді як Exploit DB пропонує експлойти для практичного тестування.

Проведений аналіз OWASP Top 10 і баз даних відомих вразливостей підкреслює необхідність систематичного моніторингу загроз для підвищення захищеності вебсайтів. Ці ресурси є основою для розробки запропонованого методу та засобу в наступних розділах роботи.

1.6 Постановка завдання дослідження

Аналіз методів дослідження безпеки вебсайтів показав, що сучасні інструменти тестування на проникнення та сканування вразливостей здебільшого орієнтовані на ручне тестування. Вони є фрагментарними і потребують значних зусиль для виявлення загроз, що знижує їхню ефективність.

Виникає потреба у створенні нового підходу до аналізу безпеки вебсайтів, який зменшить залежність від ручного тестування і забезпечить комплексне виявлення вразливостей. Розробка має поєднувати автоматизоване сканування з інтеграцією кількох баз даних. Сканер повинен виявляти вразливості клієнтської і серверної частин вебзастосунків, включаючи помилки в обробці даних, щоб забезпечити комплексне врахування потенційних загроз.

Для реалізації методу необхідно розробити математичну модель, яка описуватиме процес аналізу безпеки, а також алгоритми сканування, що оптимізують виявлення вразливостей. Програмний засіб має бути гнучким, здатним адаптуватися до різних типів вебзастосунків, включаючи системи з динамічним вмістом. Також застосунок повинен підтримувати динамічне оновлення бази вразливостей для оперативного реагування на нові загрози і створювати стандартизовані звіти з детальними рекомендаціями щодо усунення проблем.

Очікуваним результатом є створення методу та засобу, які підвищать точність і швидкість виявлення вразливостей, зменшать кількість помилкових спрацьовувань і спростять роботу фахівців із кібербезпеки.

1.7 Висновки з розділу

У даному розділі проведено аналіз методів дослідження безпеки вебсайтів, що охоплював нормативно-правову базу, структуру вебзастосунків, засоби тестування на проникнення, сканери вразливостей та бази даних відомих загроз.

Аналіз нормативно-правової бази показав необхідність отримання письмового дозволу для легального тестування безпеки вебсайтів, щоб уникнути юридичних ризиків. Дослідження структури вебзастосунків виявило вразливості в клієнтській і серверній частині, що вимагають комплексного підходу до їх виявлення та усунення.

Аналіз інструментів тестування на проникнення показав, що вони переважно орієнтовані на ручне тестування, є фрагментарними та потребують значних зусиль для виявлення загроз. Сканери вразливостей частково

автоматизують процес, але мають обмеження через високе навантаження на ресурси та схильність до помилкових результатів.

Аналіз баз даних вразливостей показав, що одна база не містить достатньо даних, тому необхідно інтегрувати кілька джерел, які надають інформацію від описів до рекомендацій щодо виправлення. Дослідження стандарту OWASP Top 10 підкреслило важливість захисту вебзастосунків.

Проведений аналіз дозволив визначити вимоги до розробки методу та засобу, які забезпечать комплексне виявлення вразливостей, зменшать залежність від ручного тестування, інтегруватимуть кілька баз даних і підвищать ефективність пошуку вразливостей на вебресурсах.

2 МЕТОД АНАЛІЗУ БЕЗПЕКИ ВЕБСАЙТІВ

2.1 Узагальнений математичний опис завдання

Для реалізації методу аналізу безпеки вебсайтів, який забезпечує комплексне виявлення вразливостей клієнтської та серверної частин, необхідно розробити математичну модель, що описує процес сканування та оцінки загроз. Опис має враховувати особливості вебзастосунків, інтеграцію кількох баз даних вразливостей і оптимізувати процес для зменшення залежності від ручного тестування.

У межах цієї моделі вебсайт розглядається як сукупність взаємопов'язаних компонентів, що формують єдину інформаційну систему. До складу таких компонентів належать серверна частина, клієнтська частина, база даних, API та логіка їх взаємодії. Кожен з елементів потенційно містить вразливості, які можуть бути використані для несанкціонованого доступу, викрадення або модифікації даних.

Множина всіх елементів системи, що підлягають перевірці, позначається як:

$$X = \{x_1, x_2, \dots, x_n, \} \tag{2.1}$$

де x_i – окремий компонент вебсайту, який аналізується під час процесу сканування. Кожен елемент цієї множини може мати власні характеристики наприклад, бібліотеки які використовуються, конфігураційні файли чи API запити.

Результатом аналізу є множина виявлених вразливостей V, яка є підмножиною множини всіх відомих вразливостей V_{all}:

$$V \subseteq V_{all} \tag{2.2}$$

Множина V_{all} формується на основі бази відомих уразливостей, яка містить записи про типові помилки конфігурації, логічні помилки, вразливості OWASP Top 10.

Кожна знайдена вразливість $v_i \in V$ характеризується певним рівнем критичності m_i, що визначається відповідно до впорядкованої множини:

$$M = \{Low, Medium, Hight, Critical\} \quad (2.3)$$

Ця шкала базується на загальноприйнятому стандарті CVSS [68], який використовується у світовій практиці для оцінки небезпеки вразливостей. Критерії розподілу базуються на таких параметрах, як вплив на конфіденційність, цілісність і доступність даних, а також складність експлуатації.

Для знаходження вразливостей використовується множина баз даних, які містять інформацію для аналізу:

$$S = \{D_V, D_F\} \quad (2.4)$$

де D_V – база відомих вразливостей, що містить опис, сигнатури, умови виникнення та способи виявлення;

D_F – база даних для фазингу, що зберігає набори тестових даних, параметри запитів і сценарії для виявлення неочевидних або логічних помилок у поведінці системи.

Для підтримки актуальності бази вразливостей виконується її синхронізація з відкритими джерелами вразливостей:

$$update: S = S' \quad (2.5)$$

Основний процес методу полягає у перетворенні множини вхідних компонентів X у множину знайдених вразливостей V з урахуванням даних, що зберігаються у внутрішніх базах:

$$analyze: X \times S \to V \quad (2.6)$$

У ході цього процесу здійснюється сканування компонентів, виконання тестових запитів, перевірка відповідей сервера, а також пошук вразливих бібліотек.

Процес фазингу відповідає за автоматичну відправку тестових даних і використовується для дослідження реакції системи на нестандартні вхідні параметри:

$$fuzz: D_F \times X \rightarrow V \tag{2.7}$$

Математична модель для аналізу безпеки вебсайтів має наступний вигляд:

$$\{X, V, S, M, analyze(X, S), fuzz(D_F, X), update(S)\} \tag{2.8}$$

Ця модель забезпечує структурований підхід до аналізу безпеки вебсайтів, дозволяючи виявляти вразливості шляхом сканування компонентів і використання баз даних. Вона автоматизує процес, зменшує залежність від ручного тестування та підвищує ефективність завдяки синхронізації з актуальними джерелами вразливостей і застосуванню фазингу для пошуку нестандартних помилок.

2.2 Узагальнене представлення методу

Розроблений метод аналізу безпеки вебсайтів базується на комплексному підході до виявлення вразливостей клієнтської та серверної частин вебзастосунків. Він поєднує автоматизоване сканування з інтеграцією баз даних загроз, що зменшує потребу в ручному втручанні та підвищує точність виявлення завдяки послідовному та систематичному аналізу всіх компонентів системи. Метод реалізується через п'ять послідовних етапів, кожен з яких виконує функції в процесі тестування безпеки вебсайтів.

Перший етап включає ініціалізацію та синхронізацію баз даних, що забезпечує актуальність інформаційної бази для подальшого аналізу. Виконується синхронізація з відкритими джерелами вразливостей, таких як CVE та NVD, що гарантує наявність актуальної інформації про відомі вразливості. Оновлюються сигнатури вразливостей та набори тестових даних для фазингу.

Другий етап призначений для розвідки та аналізу структури вебсайту через пасивне дослідження без активного втручання в його роботу. Аналізується

HTML-структура сторінок, DOM структура, файли sitemap.xml та robots.txt для виявлення всіх доступних ресурсів та прихованих директорій. Досліджуються HTTP-заголовки для ідентифікації використовуваних серверних технологій, фреймворків та політик безпеки. Виявляються форми введення даних, API, версії бібліотек та потенційні точки входу для атак відповідно до класифікації OWASP Top 10.

Третій етап передбачає статичний аналіз та зіставлення виявлених компонентів з базою відомих вразливостей без виконання активних тестів. Перевіряються версії програмного забезпечення, бібліотек та фреймворків на наявність вразливостей. Аналізуються HTTP-заголовки безпеки щодо відсутніх або неправильно налаштованих механізмів захисту. Перевіряється клієнтський JavaScript для виявлення потенційних векторів XSS, небезпечних функцій і можливих витоків конфіденційних даних. Статичний аналіз дозволяє швидко ідентифікувати очевидні проблеми без ризику впливу на роботу системи.

Четвертий етап реалізує динамічний аналіз та фазинг через активне тестування шляхом надсилання спеціально сформованих запитів до вебзастосунків. Проводиться цілеспрямована перевірка на наявність SQL-ін'єкцій, різних типів XSS, IDOR, CSRF, XXE, SSRF, path traversal та command injection. Виконується фазинг - автоматична генерація нестандартних вхідних даних для виявлення логічних помилок та вразливостей. Аналізуються відповіді сервера, коди статусу та поведінка системи при нестандартних значеннях параметрів. Тестується реакція на некоректні HTTP-методи та маніпуляції з параметрами запитів. Динамічний аналіз виявляє проблеми, які неможливо знайти статичними методами.

П'ятий етап передбачає формування єдиного звіту на основі результатів статичного та динамічного аналізу. Виконується видалення дублікатів та помилкових спрацьовувань. Кожна виявлена вразливість класифікується за шкалою CVSS з присвоєнням рівня критичності Low, Medium, High або Critical. Формується детальний звіт, що містить опис вразливості, докази та рекомендації щодо усунення. Відповідно запропонований метод формалізується так:

Крок 1. Ініціалізація та синхронізація баз даних.

Крок 2. Пасивна розвідка структури вебсайту та виявлення компонентів.

Крок 3. Статичний аналіз виявлених компонентів.

Крок 4. Динамічний аналіз шляхом активного тестування та фазингу.

Крок 5. Формування звіту.

Таким чином, запропонований метод забезпечує комплексний аналіз безпеки вебсайту. Даний підхід інтегрує автоматизоване сканування та використання кількох баз даних, знижує залежність від ручного тестування та формує основу для подальшої програмної реалізації.

2.3 Формування бази вразливостей

Формування бази вразливостей є основним етапом розробки методу. База забезпечує зіставлення компонентів вебзастосунків із відомими вразливостями, що дозволяє ефективно виявляти ризики з мінімальним ручним аналізом.

Для реалізації методу аналізу безпеки вебсайтів необхідна гнучка та масштабована архітектура зберігання даних. Використано документо-орієнтовану модель на базі MongoDB [69, 70]. Такий підхід забезпечує легке оновлення структур, що постійно змінюються, та обробку неструктурованих даних.

Дані для бази отримуються з відкритих джерел, таких як National Vulnerability Database та Common Vulnerabilities and Exposures. Ці джерела надають описи вразливостей, оцінки критичності за CVSS і відомості про експлойти. Окремо створюється база для фазингу, яка заповнюється вручну на основі векторів атак із OWASP Top 10.

База даних cve_db призначена для централізованого зберігання та періодичного оновлення інформації про відомі вразливості вебтехнологій, отриманої з зовнішніх джерел. Вона складається з двох основних колекцій: vulnerabilities та affected_products. Така організація дозволяє ефективно розділити загальні характеристики загрози від її зв'язку з конкретними

програмними продуктами, що спрощує та прискорює виконання запитів під час сканування.

Колекція vulnerabilities містить публічні дані про вразливості, що дозволяє накопичувати інформацію та оновлювати інформацію з зовнішніх джерел для подальшого використання під час статичного аналізу вебсайтів. Така структура забезпечує швидкий доступ до ключових характеристик вразливостей під час сканування. Детальний опис атрибутів цієї колекції наведено в таблиці 2.1.

Таблиця 2.1 – Атрибути колекції vulnerabilities

Поле	Тип даних	Опис
_id	ObjectId	Первинний ключ, автоматично згенерований MongoDB.
cve_id	String	Унікальний ідентифікатор вразливості.
description	String	Повний текстовий опис вразливості.
cvss_score	Number	Числова оцінка небезпеки за шкалою CVSS.
cvss_severity	String	Категорійний рівень небезпеки.
public_exploit	Boolean	Вказує на наявність загальнодоступного експлойту.
cwe_id	String	Ідентифікатор типу слабкості за CWE.
source	String	Джерело інформації про вразливість.
recommendations	Array	Масив URL-адрес з посиланнями на рекомендації.
published_date	Date	Дата публікації інформації про CVE.
last_modified_date	Date	Дата останнього оновлення запису CVE.

Як видно з таблиці 2.1, структура колекції vulnerabilities забезпечує швидкий доступ до всіх ключових характеристик вразливостей, необхідних для автоматизованого аналізу.

Для кореляції виявлених на вебсайті компонентів з конкретними вразливостями використовується колекція affected_products. Її атрибути деталізовані в таблиці 2.2.

Таблиця 2.2 – Атрибути колекції affected_products

Поле	Тип даних	Опис
_id	ObjectId	Первинний ключ, автоматично згенерований MongoDB.

Продовження таблиці 2.3

Поле	Тип даних	Опис
cve_id	String	Логічний зовнішній ключ, що посилається на cve_id.
product_name	String	Назва програмного продукту, якого стосується CVE.
vendor_name	String	Назва виробника програмного продукту.
affected_versions	String	Рядок, що описує, які саме версії є вразливими.
cpe	String	Машиночитний рядок Common Platform Enumeration.

Вміст таблиці 2.2 показує, що ця колекція забезпечує зв’язок між програмними компонентами та вразливостями, що дозволяє сканеру визначити, чи містять вразливості наявні версії програмного забезпечення.

Друга база даних, fuzzing_db, містить дані для динамічного аналізу шляхом фазингу. Вона призначена для управління власним каталогом типів вразливостей та наборами тестів для їх активного пошуку.

Колекція vulns описує типи вразливостей, на які сканер буде перевіряти вебзастосунок за допомогою фазингу. Її структура представлена в таблиці 2.3.

Таблиця 2.3 – Атрибути колекції vulns

Поле	Тип даних	Опис
_id	ObjectId	Первинний ключ, автоматично згенерований MongoDB.
title	String	Назва типу вразливості.
attack_type	String	Загальна категорія атаки.
description	String	Внутрішній опис категорії атак.
recommendations	String	Внутрішні загальні рекомендації для виправлення.
severity	String	Внутрішній рейтинг небезпеки.

Як видно з таблиці 2.3, колекція vulns групує тести за типами атак і містить інформацію, яка використовується під час фазингу для складання звітів.

Набори тестових даних зберігаються в колекції fuzzing_datasets. Кожен набір пов'язаний з певним типом вразливості з колекції vulns. Атрибути цієї колекції наведено в таблиці 2.4.

Таблиця 2.4 – Атрибути колекції fuzzing_datasets

Поле	**Тип даних**	**Опис**
_id	ObjectId	Первинний ключ, автоматично згенерований MongoDB.
target_vuln_id	ObjectId	Ідентифікатор, що пов'язує тест з типом вразливості.
risk_level	String	Рівень ризику, пов'язаний з конкретним тестом.
payloads	Array	Масив рядків (тестові навантаження).
expected_responses	Array	Масив рядків (очікувані відповіді, що свідчать про вразливість).
fuzz_parameters	Object	Вбудований об'єкт з налаштуваннями фазера.
created_by	String	Вказує автора набору тестів.
created_date	Date	Дата створення набору.

Запропонована структура баз даних забезпечує гнучкість, масштабованість та ефективність у роботі методу аналізу безпеки. Результати проектування описуваних баз даних зображені у вигляді ERD-діаграми на рисунку 2.1.

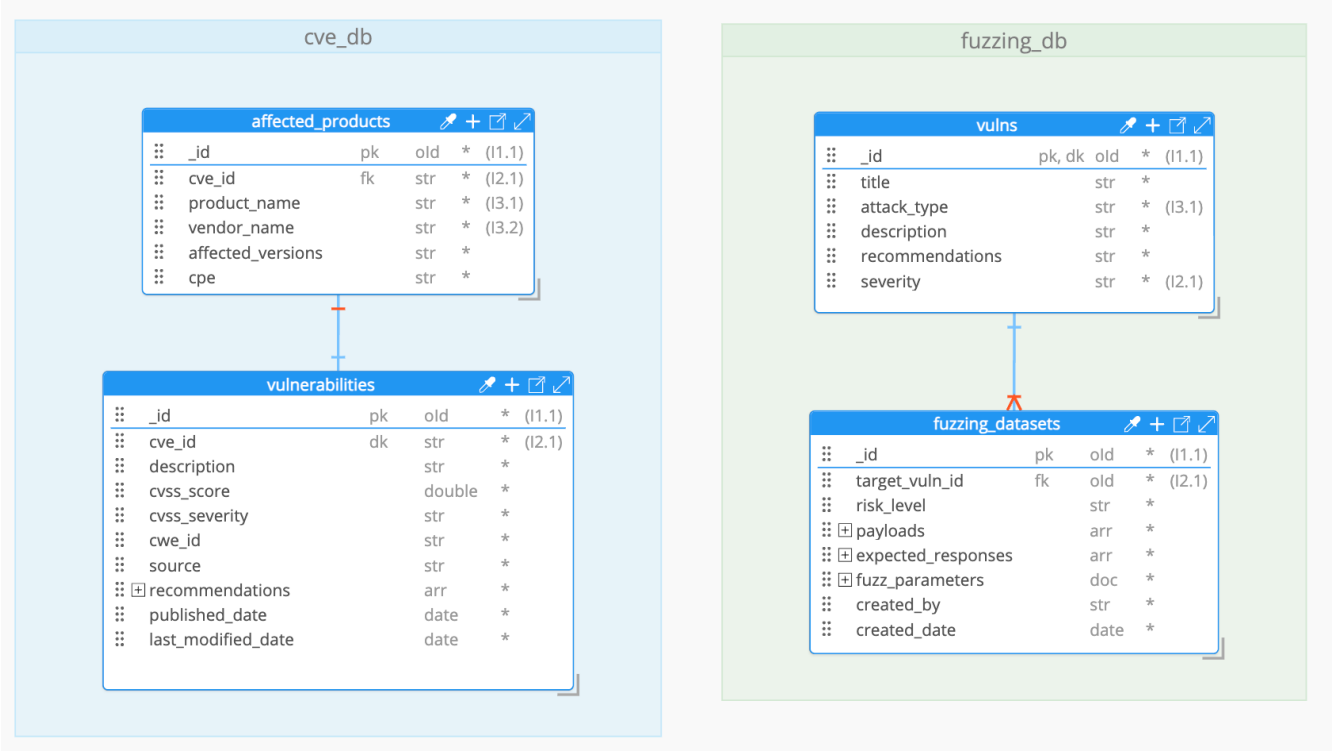


Рисунок 2.1 – ERD-діаграма бази вразливостей

Отже, розроблена база вразливостей є ключовим фундаментом для всього методу, забезпечуючи його необхідними даними для комплексного, автоматизованого та актуального аналізу безпеки вебзастосунків.

2.4 Формат звітування

Ключовим результатом роботи розробленого методу є надання чіткого, структурованого звіту про вразливості, які було виявлено. Розроблений метод передбачає генерацію звітів у трьох основних форматах TXT, XML та PDF, що забезпечує комплексне охоплення різних сценаріїв використання.

Текстовий формат реалізовано з урахуванням потреб швидкого перегляду та подальшої автоматизованої обробки. Проста структура файлу дозволяє легко інтегрувати результати сканування у відомі процеси обробки даних, використовувати їх для побудови звітів у сторонніх системах та проводити швидкий аналіз без необхідності використання спеціалізованого програмного забезпечення. Крім того, TXT-формат є універсальним та сумісним з більшістю операційних систем та інструментів обробки текстових даних.

XML формат призначений для складних корпоративних середовищ, де необхідна інтеграція з системами керування вразливостями, платформами SIEM та SOAR. Використання стандартизованих XML-схем забезпечує можливість обміну даними між різними системами.

PDF звіти орієнтовані на створення структурованих документів, що забезпечують зручне представлення інформації для аналізу результатів тестування. Цей формат є особливо корисним для створення документації, що супроводжує процес усунення вразливостей та служить довідковим матеріалом для проведення майбутніх перевірок.

Структура запису про кожну вразливість у звіті є уніфікованою та повторюється для всіх виявлених вразливостей:

– Назва вразливості (CVE/Тип атаки) – основна назва проблеми, у дужках зазначається зареєстрований CVE або тип атаки для нестандартних вразливостей.

– Scope – конкретне місце або компонент системи, де виявлено вразливість, наприклад, URL, параметр запиту, заголовок, ендпоінт API або версія бібліотеки.

– Опис – детальний опис проблеми, умови її виникнення та можливий вплив на безпеку.

– Рекомендації – практичні заходи для усунення вразливості, включаючи оновлення програмного забезпечення, зміни коду чи конфігурації та застосування додаткових засобів захисту.

Для демонстрації практичної реалізації запропонованого формату на рисунку 2.2 представлено приклад макету звіту у текстовому форматі, що містить опис вразливості SQL Injection.

```
ЗВІТ ПРО СКАНУВАННЯ БЕЗПЕКИ
Ціль: https://example.com
Дата сканування: {date}
Версія сканера: 2.0

SQL Injection (CWE-89)
CVSS: HIGH

Scope:
• https://example.com/search?query=test

Description:
Вразливість дозволяє виконувати довільні SQL-запити через неналежну валідацію вхідних даних.
Може призвести до витоку конфіденційної інформації з бази даних.

Recommendation:
1. Використовувати параметризовані запити
2. Застосувати валідацію вхідних даних
3. Обмежити привілеї облікового запису БД
```

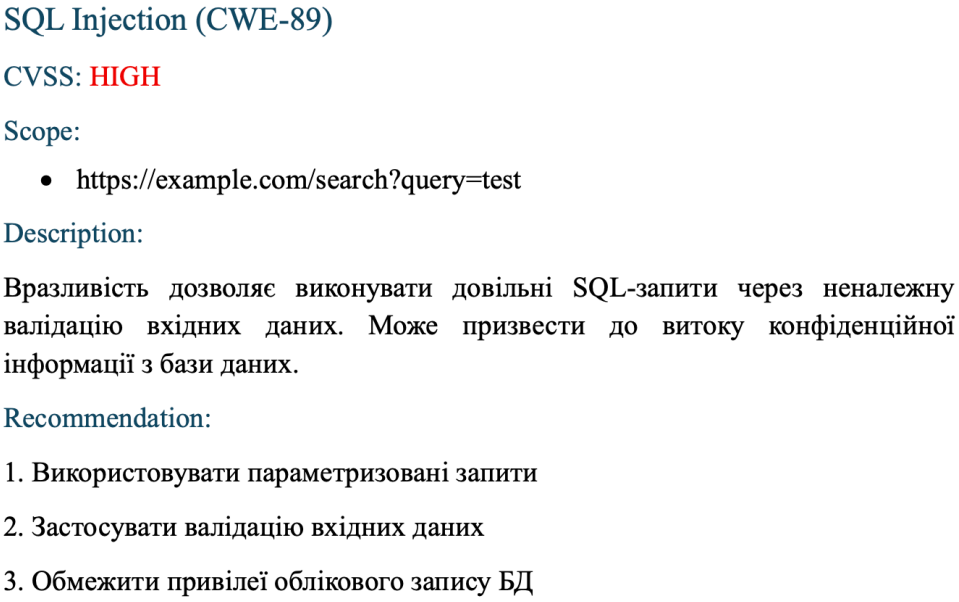
Рисунок 2.2 – Макет звіту у форматі TXT

Для забезпечення машинного читання та автоматичної обробки результатів сканування звіт генерується у XML-форматі. Приклад макету такого XML документа представлено на рисунку 2.3.

```
<security_report>
    <scan_info>
        <target_url>https://example.com</target_url>
        <scan_date>{data}</scan_date>
        <scanner_version>2.0</scanner_version>
    </scan_info>
    <vulnerabilities>
        <vulnerability>
            <name>SQL Injection (CWE-89)</name>
            <cvss_score>8.6</cvss_score>
            <cvss_severity>HIGH</cvss_severity>
            <scope>
                <item>https://example.com/search?query=test</item>
            </scope>
            <description>Вразливість дозволяє виконувати довільні SQL-запити через неналежну валідацію вхідних даних.
            Може призвести до витоку конфіденційної інформації з бази даних.</description>
            <recommendation>
                1. Використовувати параметризовані запити
                2. Застосувати валідацію вхідних даних
                3. Обмежити привілеї облікового запису БД
            </recommendation>
        </vulnerability>
    </vulnerabilities>
</security_report>
```

Рисунок 2.3 – Макет звіту у форматі XML

Для представлення результатів у вигляді структурованого документа зручного для перегляду використовується PDF формат. Приклад оформлення такого звіту показано на рисунку 2.4.


SQL Injection (CWE-89)

CVSS: HIGH

Scope:

- https://example.com/search?query=test

Description:

Вразливість дозволяє виконувати довільні SQL-запити через неналежну валідацію вхідних даних. Може призвести до витоку конфіденційної інформації з бази даних.

Recommendation:

1. Використовувати параметризовані запити
2. Застосувати валідацію вхідних даних
3. Обмежити привілеї облікового запису БД

Рисунок 2.4 - Макет звіту у форматі PDF

Запропонований підхід до форматування забезпечує стандартизацію вихідних даних, що спрощує їх аналіз. Можливість отримати звіт у різних форматах робить метод універсальним та адаптивним до різних сценаріїв використання.

2.5 Висновки з розділу

У даному розділі розроблено метод аналізу безпеки вебсайтів, який ґрунтується на комплексному підході до виявлення вразливостей клієнтської та серверної частин вебзастосунків. Проведена робота дозволила сформулювати цілісну методологію, що поєднує теоретичні основи з практичними аспектами тестування безпеки.

Розроблена математична модель формалізує процес сканування через множини компонентів системи, вразливостей та баз даних, що забезпечує структурований підхід до аналізу безпеки. Модель враховує як статичний аналіз

відомих вразливостей, так і динамічне тестування методом фазингу, що дозволяє охопити широкий спектр потенційних загроз.

Запропонована п’яти етапна методологія охоплює повний цикл тестування - від підготовки та розвідки до статичного та динамічного аналізу з подальшим формуванням звіту. Це дозволяє системно виявляти вразливості різного типу та мінімізувати залежність від ручного тестування.

Архітектура баз даних вразливостей на основі MongoDB забезпечує ефективне зберігання та обробку інформації про загрози. Спеціалізовані бази даних cve_db та fuzzing_db з чітко структурованими колекціями дозволяють інтегрувати дані з різних джерел та підтримувати їх актуальність. Використання документо-орієнтованої СУБД забезпечує гнучкість та масштабованість системи зберігання даних.

Звітування з підтримкою трьох форматів забезпечує адаптивність до різних потреб користувачів. Уніфікована структура звіту дозволяє представляти інформацію про виявлені вразливості для подальшого аналізу та усунення.

Розроблений метод забезпечує комплексний підхід до аналізу безпеки вебсайтів, що дозволяє автоматизувати процес виявлення вразливостей.

3 ЗАСІБ АНАЛІЗУ БЕЗПЕКИ ВЕБСАЙТІВ

3.1 Архітектура засобу

Архітектура засобу аналізу безпеки вебсайтів базується на принципі модульності, що забезпечує логічне розподілення функціоналу між окремими компонентами системи. Такий підхід дозволяє підвищити ефективність роботи та спрощує розширення функціоналу в майбутньому. На рисунку 3.1 представлено загальну архітектуру розробленого засобу.

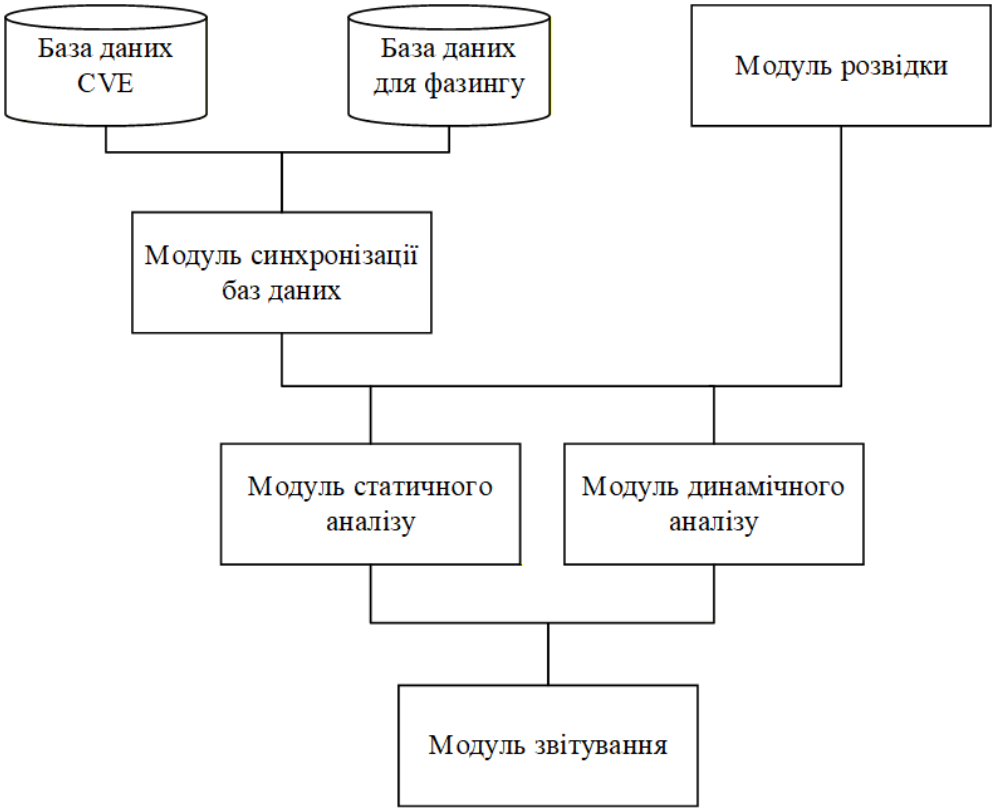


Рисунок 3.1 – Архітектурна засобу аналізу безпеки вебсайтів

Як видно з рисунку 3.1, архітектура системи включає п'ять основних модулів. Модуль синхронізації баз даних відповідає за підтримку актуальності інформації про вразливості шляхом регулярного оновлення з зовнішніх джерел. Він забезпечує наявність останніх записів CVE та пов'язаних з ними даних, що є критично важливим для статичного аналізу.

Модуль розвідки здійснює пасивний збір інформації про цільовий вебсайт, аналізуючи його структуру, використовувані технології та можливі вектори атак.

Отримані дані використовуються для подальшого аналізу іншими модулями системи.

Модуль статичного аналізу проводить безконтактне дослідження цільового застосунку шляхом аналізу вихідного коду та конфігурацій. Модуль динамічного аналізу здійснює активне тестування, імітуючи реальні атаки на вебсайт для виявлення вразливостей, що неможливо знайти статичними методами.

Всі отримані результати об'єднуються модулем звітування, який формує структуровані звіти з переліком знайдених вразливостей, їх критичності та рекомендацій щодо усунення.

Основні переваги використання такої архітектури включають:

– модульність забезпечує чітке розділення відповідальності між компонентами системи, що спрощує розробку, тестування та подальше обслуговування;

– масштабованість дозволяє незалежно розширювати або змінювати окремі модулі без впливу на інші частини системи, підвищуючи продуктивність і стійкість роботи;

– гнучкість надає можливість легко додавати нові модулі або адаптувати ті, що використовуються до змін у середовищі чи вимогах.

У результаті такого підходу архітектурне рішення забезпечує можливість подальшого розширення функціоналу, спрощує інтеграцію нових модулів та компонентів, а також підвищує гнучкість і масштабованість системи.

3.2 Узагальнений алгоритм роботи засобу

Робота засобу аналізу безпеки організована у вигляді послідовно-паралельного процесу, що забезпечує координацію всіх модулів для комплексного оцінювання безпеки вебсайту. Послідовність виконання основних етапів цього процесу відображена на рисунку 3.2.

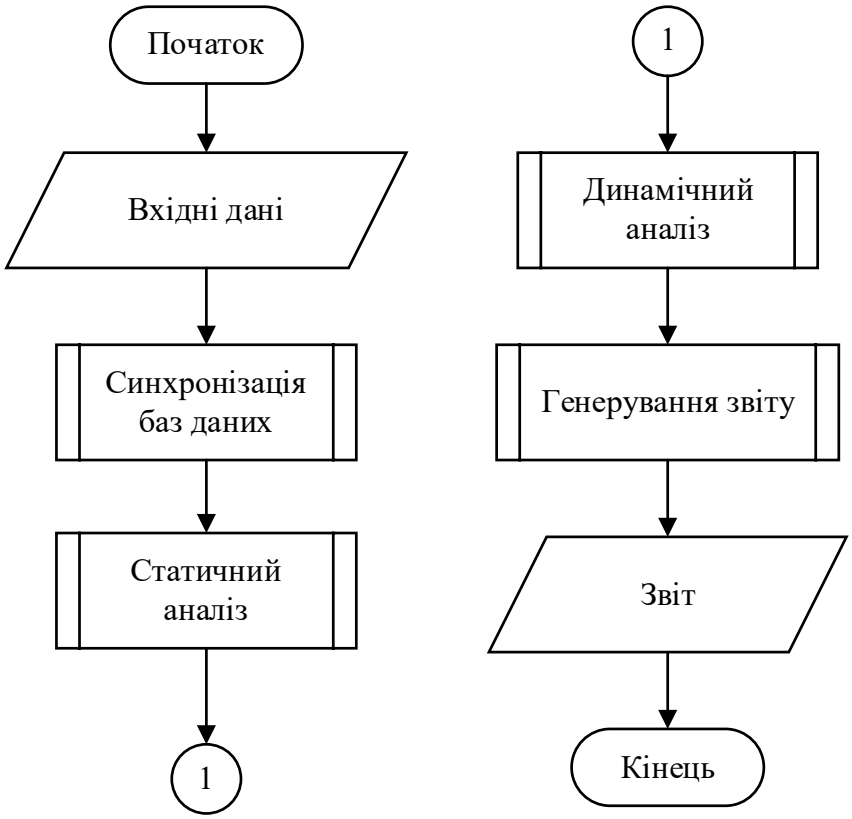


Рисунок 3.2 – Узагальнений алгоритм роботи засобу

Як видно з рисунку 3.2, алгоритм починається з процесу синхронізації баз даних, під час якого оновлюються локальні бази даних про вразливості. Актуальність цих даних є критично важливою для подальших етапів аналізу.

На наступному етапі відбувається отримання вхідних даних. На цьому етапі користувач вказує цільову URL-адресу та параметри для сканування, а система перевіряє її коректність та доступність, що є необхідною умовою для подальшого аналізу.

Далі виконується пасивна розвідка, мета якої — збір інформації про цільовий вебсайт. Модуль розвідки ідентифікує використовувані технології, аналізує структуру сайту та визначає потенційні вектори атак, формуючи основу для подальшого тестування.

Отримані дані використовуються на етапі статичного аналізу, де система зіставляє знайдені технології з актуальною базою вразливостей. Цей підхід дозволяє виявити відомі проблеми безпеки без безпосереднього втручання в роботу застосунку.

Паралельно або послідовно з цим проводиться динамічний аналіз, який імітує реальні атаки шляхом відправки спеціально сформованих запитів. Аналізуючи відповіді системи, цей модуль виявляє активні вразливості, що неможливо виявити пасивними методами.

Після завершення аналітичних етапів відбувається узагальнення та обробка результатів. Система поєднує дані з усіх модулів, усуває дублікати та оцінює загальний рівень загроз. Фінальним етапом є автоматичне формування звіту, де всі виявлені вразливості структуровані за рівнем критичності з наданням рекомендацій щодо їх усунення.

3.3 Алгоритм сканування

Алгоритм сканування є основним механізмом засобу аналізу безпеки, що реалізує три взаємопов'язані методи дослідження вебсайтів: пасивна розвідка, статичне та динамічне сканування.

Алгоритм пасивної розвідки здійснює збір інформації про цільовий вебсайт без активного втручання в його роботу. Цей метод дозволяє отримати структуру сайту, бібліотеки та потенційні вектори атаки. На рисунку 3.3 представлено алгоритму пасивного сканування.

Процес сканування базується на послідовному обході вебсайту з використанням черги URL та перевіркою кожної сторінки. Алгоритм аналізує структуру посилань, динамічно додає нові адреси до черги й уникає повторних запитів уже оброблених ресурсів. Такий підхід забезпечує повне охоплення сайту без створення зайвого навантаження.

Під час виконання збираються дані про всі виявлені сторінки, скрипти, бібліотеки, заголовки та інші елементи вебзастосунку. На основі цієї інформації формуються структуровані дані, що включають карту сайту та каталог ресурсів. Отримані результати використовуються на наступних етапах для статичного й динамічного аналізу безпеки.

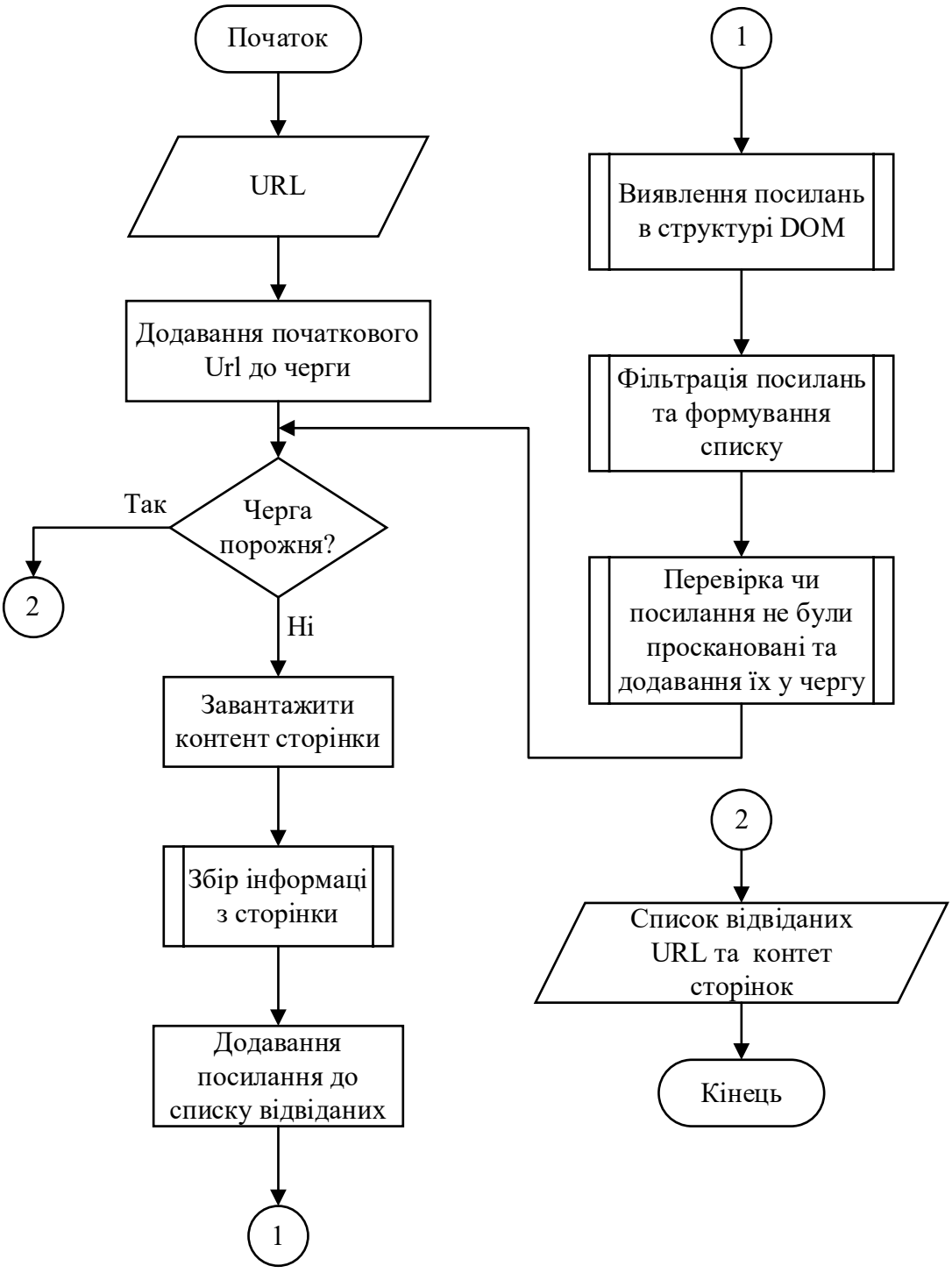


Рисунок 3.3 – Алгоритм пасивної розвідки

Алгоритм статичного аналізу здійснює дослідження вебзастосунку шляхом аналізу отриманих HTTP-запитів, відповідей сервера та структури вебресурсів без активного впливу на роботу сайту. Такий підхід дозволяє виявляти потенційні вразливості на основі зібраної інформації. На рисунку 3.4 представлено алгоритм статичного аналізу.

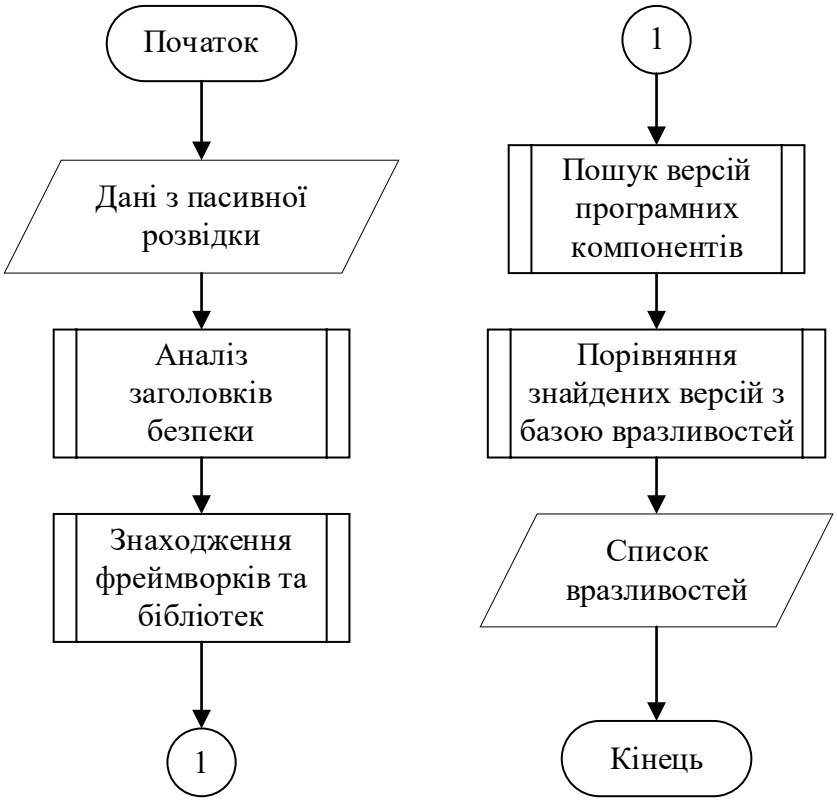


Рисунок 3.4 – Алгоритм статичного аналізу

Як видно з рисунку 3.4, алгоритм статичного аналізу починається з обробки даних, отриманих на етапі пасивного сканування вебзастосунку. На основі зібраної інформації про використовувані технології та їх версії здійснюється пошук вразливих компонентів і бібліотек шляхом зіставлення з актуальними базами вразливостей.

Такий підхід дозволяє виявляти відомі вразливості без активного втручання в роботу вебзастосунку. У результаті формується перелік виявлених вразливостей, який передається до модуля звітування.

Алгоритм динамічного сканування здійснює активне тестування вебсайту шляхом відправки спеціально сформованих запитів та подальшого аналізу відповідей сервера. У процесі роботи алгоритму перевіряється поведінка вебзастосунку за різних вхідних даних з метою виявлення можливих вразливостей.

На рисунку 3.5 представлено алгоритм динамічного сканування.

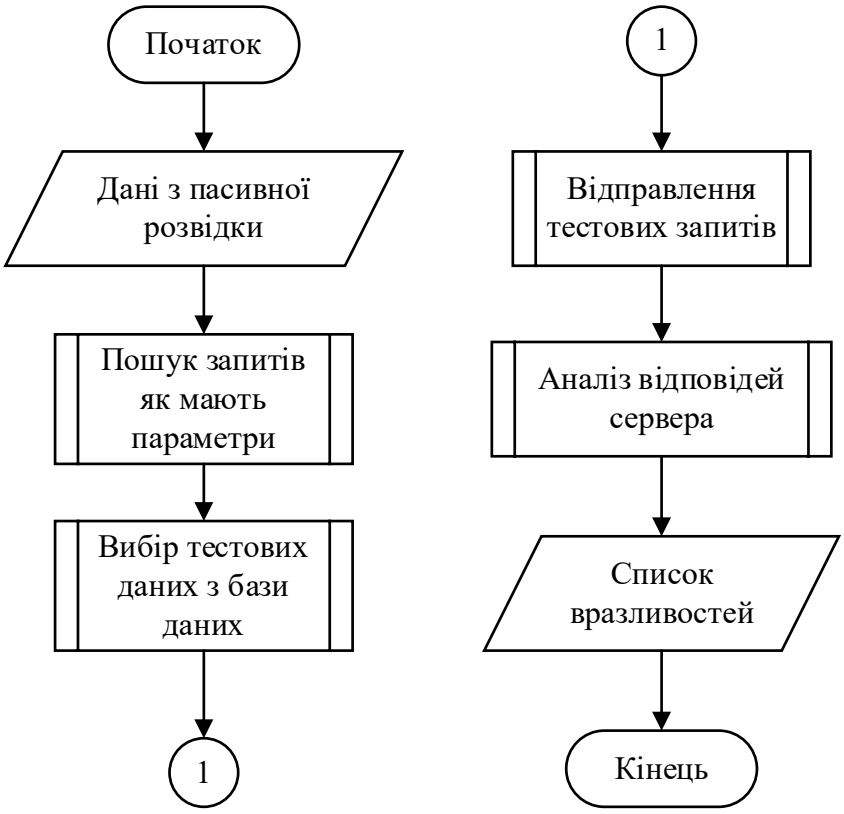


Рисунок 3.5 – Алгоритм динамічного сканування

Як видно з рисунку 3.5, алгоритм динамічного сканування починається з отримання даних з пасивної розвідки. На даному етапі ідентифікуються точки взаємодії з користувачем, такі як параметри запитів, поля форм та ендпоінти API.

Для кожної знайденої точки з бази даних фазингу завантажуються спеціальні тестові корисні навантаження, призначені для виявлення конкретних класів вразливостей.

Після відправки сформованих запитів на сервер проводиться аналіз отриманих відповідей, зокрема перевіряються коди статусів HTTP, вміст HTML та інші метрики. Відповіді, що свідчать про аномальну поведінку, фіксуються та класифікуються за типом і рівнем ризику та додаються до остаточного списку виявлених вразливостей.

Таким чином, послідовне застосування пасивної розвідки, статичного та динамічного аналізу забезпечує комплексний підхід до оцінювання безпеки вебсайту.

3.4 Алгоритм формування звіту

Алгоритм формування звіту є завершальним етапом роботи засобу аналізу безпеки, що забезпечує систематизацію та представлення результатів дослідження. Основним завданням цього алгоритму є об'єднання даних, отриманих від усіх модулів сканування, у єдиний структурований документ з класифікацією виявлених вразливостей. На рисунку 3.6 представлено алгоритм формування звіту.

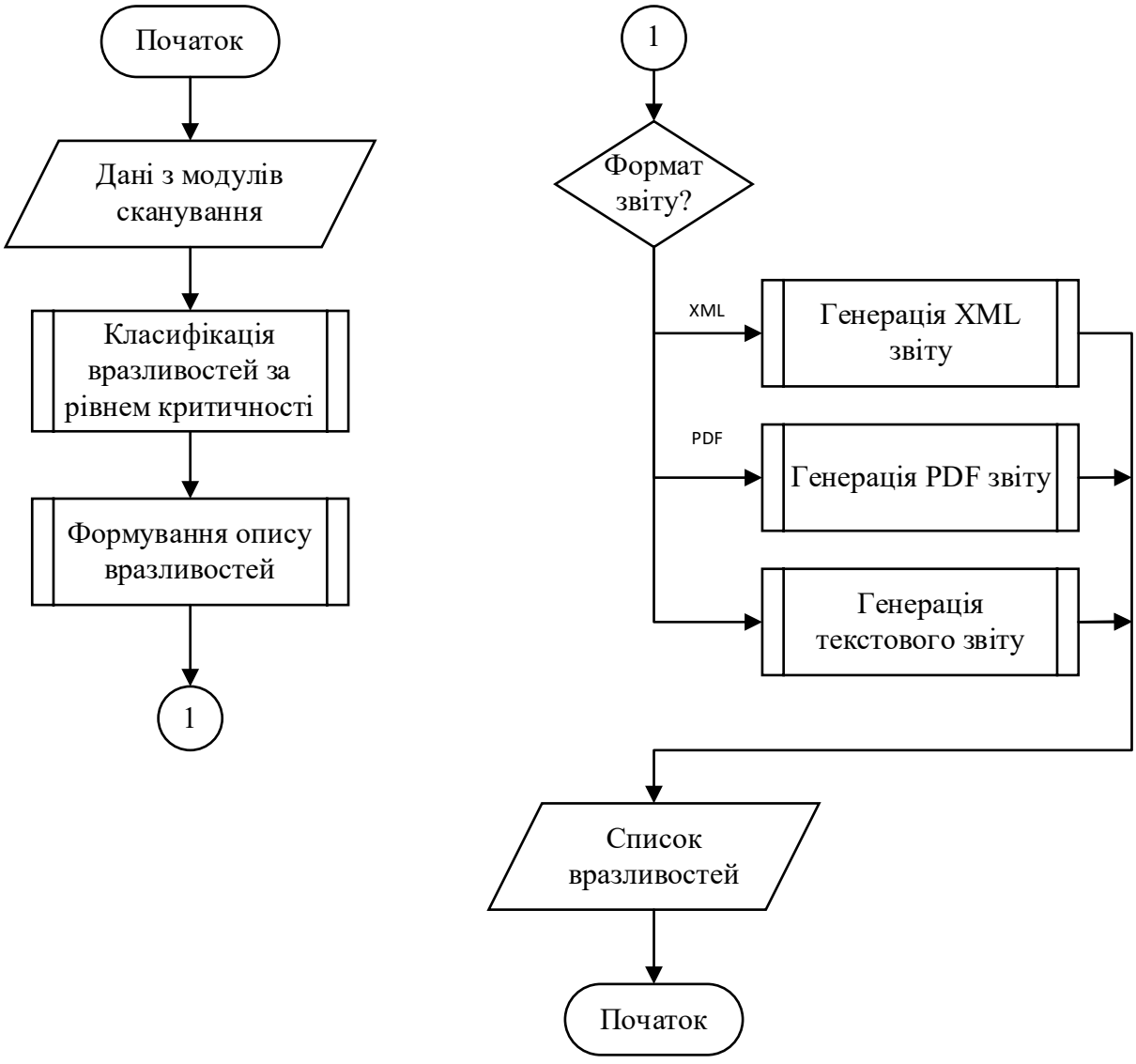


Рисунок 3.6 – Алгоритм формування звіту

Як видно з рисунку 3.6, процес формування звіту починається зі збору результатів з усіх модулів сканування. На цьому етапі відбувається об'єднання інформації про вразливості, виявлені під час пасивної розвідки, статичного та

динамічного аналізу. Система автоматично видаляє дублікати та формує єдиний перелік знайдених проблем.

Після об'єднання даних здійснюється сортування вразливостей за рівнем критичності відповідно до стандарту CVSS. Кожна вразливість аналізується та розподіляється за ступенем загрози. Для кожної вразливості формується опис, який включає тип загрози, місце виявлення, докази знаходження та рекомендації щодо усунення.

Фінальним етапом роботи алгоритму є генерація звіту у вибраних форматах виводу. Підтримуються три основні формати: TXT для швидкого перегляду, XML для автоматизованої обробки та PDF для зручного форматування. Для кожного формату використовуються шаблони, що забезпечують оптимальне представлення інформації з урахуванням особливостей формату.

Запропонований алгоритм автоматизує процес створення звітів, що дає змогу скоротити час на аналіз результатів тестування та надає рекомендації щодо усунення виявлених вразливостей.

3.5 Обґрунтування вибору засобів розробки

Вибір інструментів розробки для програмного засобу аналізу безпеки вебсайтів ґрунтувався на комплексі вимог, що випливають із специфіки завдання. Основними вимогами до засобів розробки була необхідність забезпечити високу продуктивність при обробці великої кількості мережевих запитів, що особливо важливо для масштабного сканування вебресурсів. Крім того, критично важливим було мінімізувати ймовірність появи вразливостей у програмному коді, оскільки інструмент аналізу безпеки сам повинен бути захищеним від різних видів атак. Також враховувалася зручність розробки та подальшого вдосконалення, включаючи наявність сучасної екосистеми бібліотек та інструментів для реалізації складних алгоритмів аналізу.

Для вибору оптимальної технології було проведено порівняльний аналіз сучасних мов програмування, які використовуються для розробки подібних

систем. Критерії порівняння включали продуктивність, захищеність пам'яті, можливості паралельного виконання, розвиненість екосистеми та складність освоєння. Результати аналізу представлено в таблиці 3.1.

Таблиця 3.1 – Порівняльний аналіз мов програмування

Мова	**Продуктивність**	**Безпека пам’яті**	**Екосистема**	**Складність навчання**
Rust	Висока завдяки компіляції в машинний код і оптимізації компілятора	Вбудовані механізми контролю доступу до пам’яті на етапі компіляції	Активний розвиток бібліотек	Висока через специфічну модель роботи з даними
C++	Дуже висока швидкодія та широкий діапазон оптимізацій	Потребує ручного керування пам’яттю, що ускладнює розробку	Широка база інструментів і бібліотек	Дуже висока через складність синтаксису
Java	Висока продуктивність завдяки JIT-компіляції	Автоматичне керування пам’яттю, стабільна модель виконання	Розвинений корпоративний стек технологій	Середня, вимагає розуміння роботи JVM
Go	Висока швидкодія	Автоматичне керування пам’яттю, безпечна модель роботи з ресурсами	Підтримка мережевих та інфраструктурних бібліотек	Помірна, мова мінімалістична та структурована
Python	Низька через інтерпретацію	Автоматичне керування пам’яттю	Велика кількість бібліотек	Низька, зручна для швидкого написання коду

Як видно з таблиці 3.1, мова програмування Rust поєднує високу продуктивність із вбудованими механізмами контролю пам’яті на етапі компіляції. Хоча C++ демонструє максимальну швидкодію, Rust забезпечує стабільне й ефективне виконання, що особливо важливо для мережевих застосунків, які обробляють великі обсяги даних, оскільки це дозволяє запобігати критичним помилкам, пов’язаним із керуванням пам’яттю.

Порівняно з Java, Rust пропонує вищу продуктивність за рахунок відсутності віртуальної машини та прямого доступу до системних ресурсів. На відміну від Python, Rust компілюється в машинний код, що забезпечує значно

вищу швидкість виконання операцій аналізу. Хоча Go демонструє високу продуктивність, Rust має переваги у більш строгій системі типів та відсутності збирача сміття.

На основі проведеного аналізу, представленого в таблиці 3.1, для реалізації проекту було обрано мову програмування Rust. Архітектура засобу ґрунтується на сучасних бібліотеках екосистеми Rust. Для мережевої взаємодії використовується асинхронна бібліотека Reqwest, яка забезпечує ефективну обробку великої кількості одночасних HTTP-запитів. Аналіз HTML-вмісту реалізовано за допомогою бібліотеки Scraper, що надає високошвидкісний парсинг вебсторінок. Серіалізація даних виконується через бібліотеку Serde, яка забезпечує роботу з форматами JSON та XML.

Для зберігання інформації про вразливості розглядалися різні типи систем керування базами даних. Реляційні СУБД, такі як PostgreSQL чи MySQL, пропонують строгу схему даних та надійні транзакції, але вимагають чітко визначеної структури, що ускладнює роботу з напівструктурованими даними про вразливості. Системи типу ключ-значення, наприклад Redis, забезпечують високу швидкість доступу, але мають обмежені можливості для виконання складних запитів та аналізу даних.

MongoDB було обрано через низку переваг, які відповідають вимогам. Документо-орієнтована модель даних дозволяє зберігати інформацію про вразливості у вигляді гнучких документів, що особливо важливо при роботі з даними різних форматів і джерел. Гнучка схема зберігання дає можливість легко адаптувати структуру бази даних до змін у джерелах інформації про загрози. MongoDB ефективно працює з напівструктурованими даними, що дозволяє зберігати різноманітні атрибути вразливостей без попереднього визначення всіх полів. Система підтримує горизонтальне масштабування, що забезпечує можливість обробки великих обсягів даних про вразливості.

Наявність стабільних драйверів для мови програмування Rust спрощує інтеграцію бази даних з основним засобом. MongoDB також забезпечує потужні механізми індексації та агрегації даних, що дозволяє швидко виконувати складні

запити для пошуку та аналізу вразливостей під час сканування вебсайтів. Асинхронне виконання завдань реалізоване на основі Tokio, що забезпечує високу пропускну здатність при обробці паралельних запитів до вебресурсів.

Таким чином, обраний набір інструментів розробки на основі проведеного порівняльного аналізу забезпечує оптимальний баланс між продуктивністю, захищеністю та зручністю розробки, що робить його найбільш прийнятним рішенням для реалізації засобу аналізу безпеки вебсайтів.

3.6 Висновки з розділу

У даному розділі було розроблено архітектуру та алгоритми роботи засобу аналізу безпеки веб сайтів. Запропонована архітектура ґрунтується на модульному підході, що включає п'ять взаємопов'язаних компонентів: синхронізацію баз даних, розвідку, статичний та динамічний аналіз, а також звітування. Така організація системи забезпечує чітке розділення функціоналу та спрощує її розширення.

Робота засобу реалізована через чітко визначену послідовність дій, яка координує всі модулі для оцінки безпеки. Алгоритм сканування поєднує три методи дослідження: пасивну розвідку для збору інформації про структуру сайту, статичний аналіз для виявлення відомих вразливостей компонентів та динамічний аналіз для активного тестування.

Завершальним етапом є алгоритм формування звіту, який автоматизує процес об'єднання результатів, класифікації вразливостей за рівнем критичності та генерації звітів у трьох форматах.

Для реалізації засобу обрано мову програмування Rust та відповідні бібліотеки, що забезпечують ефективну роботу з мережевими запитами та обробку даних.

Розроблений засіб є практичним рішенням для автоматизованого аналізу безпеки вебсайтів, який поєднує різні методи тестування та забезпечує комплексний підхід до виявлення вразливостей. Модульна архітектура дозволяє легко адаптувати систему до нових вимог і впроваджувати додаткові функції.

4 ЕКСПЕРИМЕНТАЛЬНІ ДОСЛІДЖЕННЯ ТА ТЕСТУВАННЯ

4.1 Модульне тестування

Для оцінки роботи розробленого засобу аналізу безпеки вебсайтів було проведено комплексне модульне тестування його окремих компонентів. Тестування розпочалося з перевірки баз даних, оскільки вони становлять основу для функціонування всього інструменту. Першим було протестовано модуль роботи з базою даних CVE, який відповідає за зберігання та управління інформацією про відомі вразливості.

Для прикладу варто розглянути один з ключових тестів цього модулю. На рисунку 4.1 представлено код модульного тесту, що перевіряє функціонал завантаження та розпакування архіву з базою даних CVE.

```rust
#[tokio::test]
async fn fetch_and_extract_cve() {
    use std::time::Instant;

    let tmp : TempDir  = tempdir().expect( msg: "tempdir");
    let dest : PathBuf  = tmp.path().join( path: "cvelistV5");

    let t0 : Instant  = Instant::now();
    let extracted : PathBuf  = download_cve::fetch_and_extract(download_cve::DEFAULT_CVELIST_URL, &dest)
        .await : Result<PathBuf, Box<...>>
        .expect( msg: "fetch_and_extract ok");
    let first_elapsed : Duration  = t0.elapsed();

    assert!(extracted.exists(), "extracted dir must exist");
    let marker : PathBuf  = dest.join( path: ".extracted");
    assert!(marker.exists(), "marker .extracted must be created");


    let inner : PathBuf  = download_cve::find_inner_root(&dest).expect( msg: "inner directory should exist after unzip");
    assert!(inner.is_dir(), "inner root must be a directory");
    let mut has_entries : bool  = false;
    if let Ok(mut rd : ReadDir ) = std::fs::read_dir(&inner) {
        has_entries = (&mut rd ).next().is_some();
    }
    assert!(has_entries, "inner directory should contain files or subdirectories");
    println!("CVE download+extract time (first run): {:?}", first_elapsed);
}
```

Рисунок 4.1 – Код модульного тесту завантаження бази даних CVE

Як показано на рисунку 4.1, тест fetch_and_extract_cve виконує комплексну перевірку роботи модуля завантаження CVE. Він включає створення тимчасової директорії для зберігання даних, завантаження архіву з офіційного джерела CVE,

розпакування отриманих даних та перевірку коректності виконаних операцій. Перевірка включає контроль наявності розпакованої директорії, маркера успішного виконання, а також переконується, що внутрішня директорія містить файли чи піддиректорії.

Для перевірки роботи з архівами NVD було реалізовано додатковий тест, який виконує перевірку завантаження даних за кілька років. На рисунку 4.2 представлено код модульного тесту, що перевіряє функціонал завантаження архівів NVD за період від 2002 року до поточного року.

```rust
#[tokio::test]
async fn fetch_and_extract_nvd_all_years() {
    use chrono::Datelike;

    let now : DateTime<Utc> = chrono::Utc::now();
    let y_to : i32 = now.year();
    let y_from : i32 = 2002;

    let tmp : TempDir = tempdir().expect( msg: "tempdir");

    for y : i32 in y_from..=y_to {
        let dest : PathBuf = tmp.path().join( path: format!("nvd-{}", y));
        let url : String = download_cve::NVD20_URL_TMPL.replace( from: "{year}", to: &format!("{}", y));

        let extracted : PathBuf = download_cve::fetch_and_extract(&url, &dest)
            .await : Result<PathBuf, Box<...>>
            .expect( msg: &format!("fetch_and_extract (NVD {}) ok", y));

        assert!(extracted.exists(), "extracted dir must exist for {}", y);
        let marker : PathBuf = dest.join( path: ".extracted");
        assert!(marker.exists(), "marker .extracted must be created for NVD {}", y);

        let json_name : String = format!("nvdcve-2.0-{}.json", y);
        let json_path : PathBuf = dest.join(&json_name);
        assert!(json_path.is_file(), "NVD JSON must exist after extraction: {}", json_name);
    }
}
```

Рисунок 4.2 – Код модульного тесту завантаження архівів NVD

Як видно з рисунку 4.2, тест fetch_and_extract_nvd_all_years виконує ітеративну перевірку завантаження та обробки архівів NVD для кожного року у вказаному діапазоні. Він автоматично визначає діапазон років на основі поточної дати, формує відповідні URL за шаблоном NVD20_URL_TMPL та перевіряє коректність розпакованих даних, включаючи наявність JSON-файлів формату nvdcve-2.0 зі структурованими даними про вразливості.

На рисунку 4.3 представлено результати виконання всіх модульних тестів для перевірки роботи з базами даних CVE та NVD.

```
running 11 tests
test cve_tests::default_out_dir_is_relative ... ok
test cve_tests::extract_cve_id_returns_id_or_none ... ok
test cve_tests::pick_cvss_prefers_v3_1_then_v3_0 ... ok
test cve_tests::first_cwe_returns_first_entry ... ok
test cve_tests::pick_description_prefers_english ... ok
test cve_tests::public_exploit_detected_by_tag_or_url ... ok
test cve_tests::collect_recommendations_prefers_marked ... ok
test cve_tests::returns_none_if_no_inner_dirs ... ok
test cve_tests::detect_inner_root_in_zip ... ok
test cve_tests::fetch_and_extract_nvd_all_years ... ok
test cve_tests::fetch_and_extract_cve has been running for over 60 seconds
test cve_tests::fetch_and_extract_cve ... ok

test result: ok. 11 passed; 0 failed; 0 ignored; 0 measured; 0 filtered out; finished in 155.13s
```

Рисунок 4.3 – Результати модульного тестування роботи з базами даних CVE

Як показано на рисунку 4.3, успішне проходження тестів підтверджує правильність реалізації алгоритмів обробки структурованих даних про вразливості та стабільну роботу механізмів завантаження інформації з зовнішніх джерел.

Для візуалізації обсягів обробленої інформації на рисунку 4.4 показано приклад записів з бази даних CVE в MongoDB.

```
cve_db> db.vulnerabilities.find({}, { _id: 0 }).sort({ cve_id: -1 }).limit(2)
[
  {
    cve_id: 'CVE-2025-9999',
    cwe_id: 'CWE-940',
    description: 'Some payload elements of the messages sent between two stations in a networking architecture are not properly checked on the receiving station allowing an attack
er to execute unauthorized commands in the application.',
    last_modified_date: ISODate('2025-10-31T16:47:55.704Z'),
    public_exploit: false,
    published_date: ISODate('2025-09-05T16:41:01.957Z'),
    recommendations: [ 'https://www.pcvue.com/security/#SB2025-4' ],
    source: 'cvelistV5'
  },
  {
    cve_id: 'CVE-2025-9998',
    cwe_id: 'CWE-754',
    description: 'The sequence of packets received by a Networking server are not correctly checked.\n' +
      '\n' +
      'An attacker could exploit this vulnerability to send specially crafted messages to force the application to stop.',
    last_modified_date: ISODate('2025-10-31T16:47:45.883Z'),
    public_exploit: false,
    published_date: ISODate('2025-09-05T16:40:13.645Z'),
    recommendations: [ 'https://www.pcvue.com/security/#SB2025-4' ],
    source: 'cvelistV5'
  }
]
cve_db>  db.vulnerabilities.countDocuments()
318852
```

Рисунок 4.4 – Приклад записів CVE в MongoDB

Згідно з рисунком 4.4, база даних містить понад 300 000 записів про вразливості, кожен з яких включає: ідентифікатор CVE, опис, оцінку CVSS, рівень критичності, інформацію про наявність експлойтів та посилання на

рекомендації щодо усунення. Така організація даних дозволяє ефективно виконувати пошук та аналіз вразливостей під час статичного сканування.

Наступним етапом було протестовано процес ініціалізації бази даних для фазингу, який відповідає за попереднє заповнення тестовими наборами для активного сканування. Цей процес є ключовим для динамічного аналізу вебсайтів, оскільки забезпечує наявність шаблонів атак для виявлення різних типів вразливостей. На рисунку 4.5 представлено результати тестування завантаження тестових наборів.

```
running 3 tests
test fuzzing_tests::defaults_sets_are_non_empty ... ok
test fuzzing_tests::merge_deduplicates_and_sorts ... ok
test fuzzing_tests::download_all_fuzzing_datasets ... ok

test result: ok. 3 passed; 0 failed; 0 ignored; 0 measured; 0 filtered out; finished in 7.16s
```

Рисунок 4.5 – Результати модульного тестування завантаження тестових наборів

Як видно з рисунку 4.5, тестування підтвердило завантаження тестових наборів для різних типів атак та правильне об'єднання та сортування тестових даних.

Для перевірки роботи пасивної розвідки було протестовано відповідний модуль, який відповідає за збір інформації про структуру та технології вебсайтів. Цей модуль є основою для подальшого аналізу, оскільки забезпечує необхідні вхідні дані для інших компонентів системи. На рисунку 4.6 представлено результати тестування основних функцій модуля пасивної розвідки.

```
running 12 tests
test cravler_tests::base::validate_target_extracts_headers_and_meta ... ok
test cravler_tests::base::validate_extracts_generator_from_variants ... ok
test cravler_tests::base::validate_handles_missing_headers_and_missing_meta ... ok
test cravler_tests::signatures::detects_nothing_on_empty_response ... ok
test cravler_tests::base::passive_recon_respects_max_depth ... ok
test cravler_tests::base::passive_recon_ignores_non_http_links ... ok
test cravler_tests::headers::crawler_sends_custom_headers_and_cookies ... ok
test cravler_tests::signatures::detects_technologies_from_html_only ... ok
test cravler_tests::signatures::detects_technologies_from_headers_only ... ok
test cravler_tests::signatures::crawler_follows_links_with_depth ... ok
test cravler_tests::base::passive_recon_crawls_same_origin_and_derives_vectors ... ok
test cravler_tests::signatures::detects_technologies_versions_and_cpes ... ok

test result: ok. 12 passed; 0 failed; 0 ignored; 0 measured; 0 filtered out; finished in 0.14s
```

Рисунок 4.6 – Результати тестування модуля пасивної розвідки

Як показано на рисунку 4.6, модуль пасивної розвідки успішно проходить тести, що охоплюють його функціонал: виявлення технологій з HTML-коду та заголовків HTTP-відповідей, аналіз версій технологій та їх ідентифікаторів, обробку посилань в межах одного джерела з дотриманням обмеження глибини обходу, а також коректну відправку спеціальних заголовків та кукі для взаємодії з вебсайтами, які вимагають автентифікації.

Далі було проведено тестування модуля статичного аналізу, який відповідає за зіставлення виявлених технологій з базою відомих вразливостей. На рисунку 4.7 представлено результати тестування основних функцій цього модуля.

```
running 5 tests
test static_tests::basic::static_analyze_returns_empty_when_no_cpes ... ok
test static_tests::cpe_edge_cases::static_analyze_early_return_without_cpes ... ok
test static_tests::basic::static_analyze_survives_no_db_and_returns_empty ... ok
test static_tests::cpe_edge_cases::static_analyze_accepts_edge_case_versions_tokens ... ok
test static_tests::cpe_edge_cases::static_analyze_survives_large_cpe_list has been running for over 60 seconds
test static_tests::cpe_edge_cases::static_analyze_survives_large_cpe_list ... ok

test result: ok. 5 passed; 0 failed; 0 ignored; 0 measured; 0 filtered out; finished in 100.98s
```

Рисунок 4.7 – Результати тестування модуля статичного аналізу

Як видно з рисунку 4.7, модуль статичного аналізу успішно обробляє різні сценарії роботи, включаючи роботу з відсутністю даних CPE, змішаними токенами CPE та великими списками ідентифікаторів. Він коректно повертає порожні результати при відсутності збігів з базою вразливостей та стійко працює при тимчасовій недоступності бази даних.

Фінальним етапом було протестовано модуль динамічного аналізу, який здійснює активне тестування вебсайтів через відправку спеціально підготовлених запитів. На рисунку 4.8 представлено результати тестування основних функцій цього модуля.

```
running 4 tests
test dynamic_tests::dynamic_analyze_returns_empty_on_safe_pages ... ok
test dynamic_tests::dynamic_analyze_respects_skip_paths ... ok
test dynamic_tests::dynamic_analyze_respects_headers_and_cookies ... ok
test dynamic_tests::dynamic_analyze_scans_all_statuses_including_4xx_5xx ... ok

test result: ok. 4 passed; 0 failed; 0 ignored; 0 measured; 0 filtered out; finished in 1.51s
```

Рисунок 4.8 – Результати тестування модуля динамічного аналізу

Як показано на рисунку 4.8, модуль динамічного аналізу демонструє коректну роботу з різними типами сторінок, здатність дотримуватися налаштувань пропуску певних шляхів та правильну обробку кастомних заголовків та кукі для автентифікації. Важливою особливістю є можливість сканування сторінок з будь-якими статус-кодами відповідей, включаючи помилки 4xx та 5xx, що розширює можливості виявлення потенційних загроз.

Проведені тести підтвердили правильну роботу всіх окремих модулів системи. Кожен компонент показав здатність виконувати свої функції у різних умовах. Успішне проходження модульного тестування підтверджує відповідність архітектури поставленим завданням та реалізації алгоритмів, що становить основу для функціонування всього засобу аналізу безпеки вебсайтів.

4.2 Статичне тестування безпеки засобу

Для оцінки безпеки та якості програмного коду розробленого засобу аналізу безпеки вебсайтів було проведено статичне тестування. Даний вид тестування дозволяє виявити потенційні вразливості, помилки кодування та порушення стандартів розробки без необхідності виконання програми.

Статичний аналіз є важливою складовою процесу забезпечення безпеки програмного забезпечення, оскільки дозволяє виявити проблеми на ранніх етапах розробки.

Для проведення аналізу було використано інструмент Semgrep, який спеціалізується на виявленні проблем безпеки в програмному коді. Semgrep дозволяє перевіряти код за допомогою налаштовуваних правил та шаблонів, що покривають різні категорії вразливостей, включаючи ін'єкції, небезпечне керування пам'яттю, криптографічні помилки та проблеми контролю доступу [71]. Також він дозволяє автоматизувати повторні перевірки коду після внесення змін, що спрощує підтримку безпечного програмного забезпечення. Результати проведеного сканування представлено на рисунку 4.9.

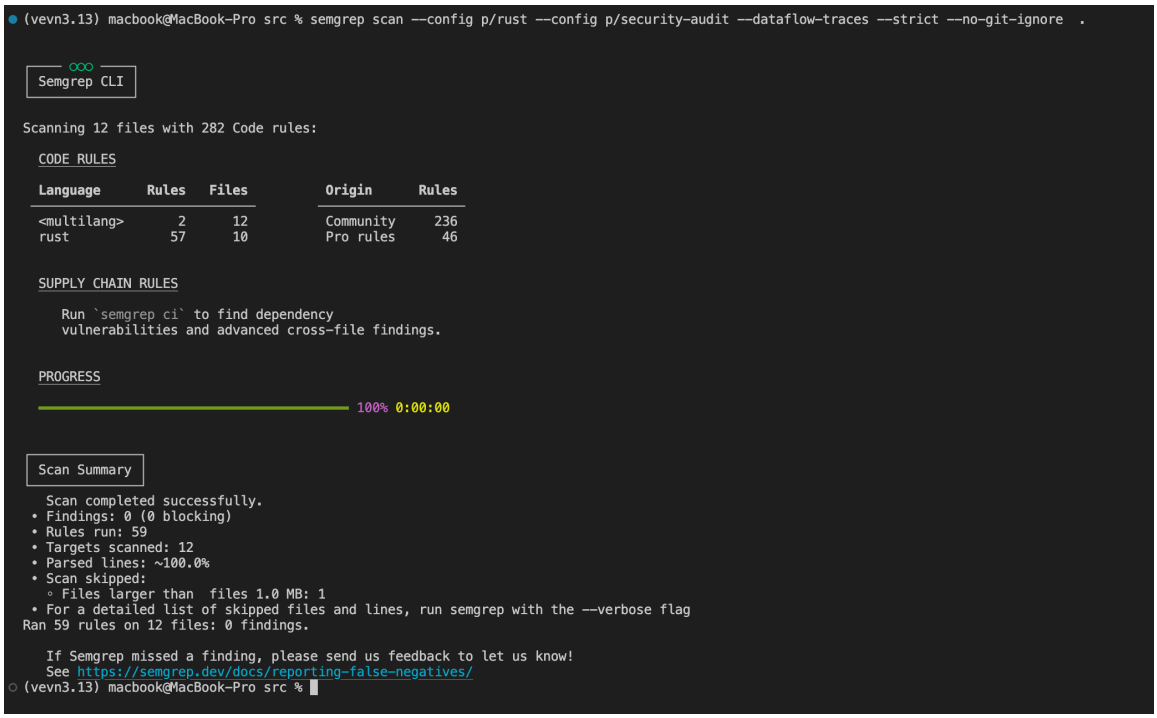


Рисунок 4.9 – Результати сканування Semgrep

Як видно з рисунку 4.9, Semgrep проаналізував 13 файлів проєкту, застосувавши 59 правил перевірки. Інструмент не знайшов вразливостей, помилок кодування чи порушень правил безпеки. Правила, застосовані під час сканування, включали 236 правил від спільноти розробників та 46 професійних правил для мови програмування Rust. Ці правила охоплюють широкий спектр потенційних проблем безпеки.

Відсутність виявлених вразливостей підтверджує належний рівень безпеки програмного коду. Для засобу аналізу безпеки вебсайтів цей результат є особливо важливим, оскільки свідчить про власну захищеність інструменту. Semgrep не виявив проблем, що демонструє дотримання принципів безпечного програмування та правильне використання можливостей мови Rust.

Для додаткової перевірки безпеки коду було використано платформу SonarQube [72]. Інструмент дозволяє оцінити якість коду та виявити потенційні проблеми безпеки на основі встановлених правил і метрик. Отримані результати допомагають визначити пріоритети для виправлення вразливостей та покращення структури коду. На рисунку 4.10 представлено результати аналізу програмного коду.

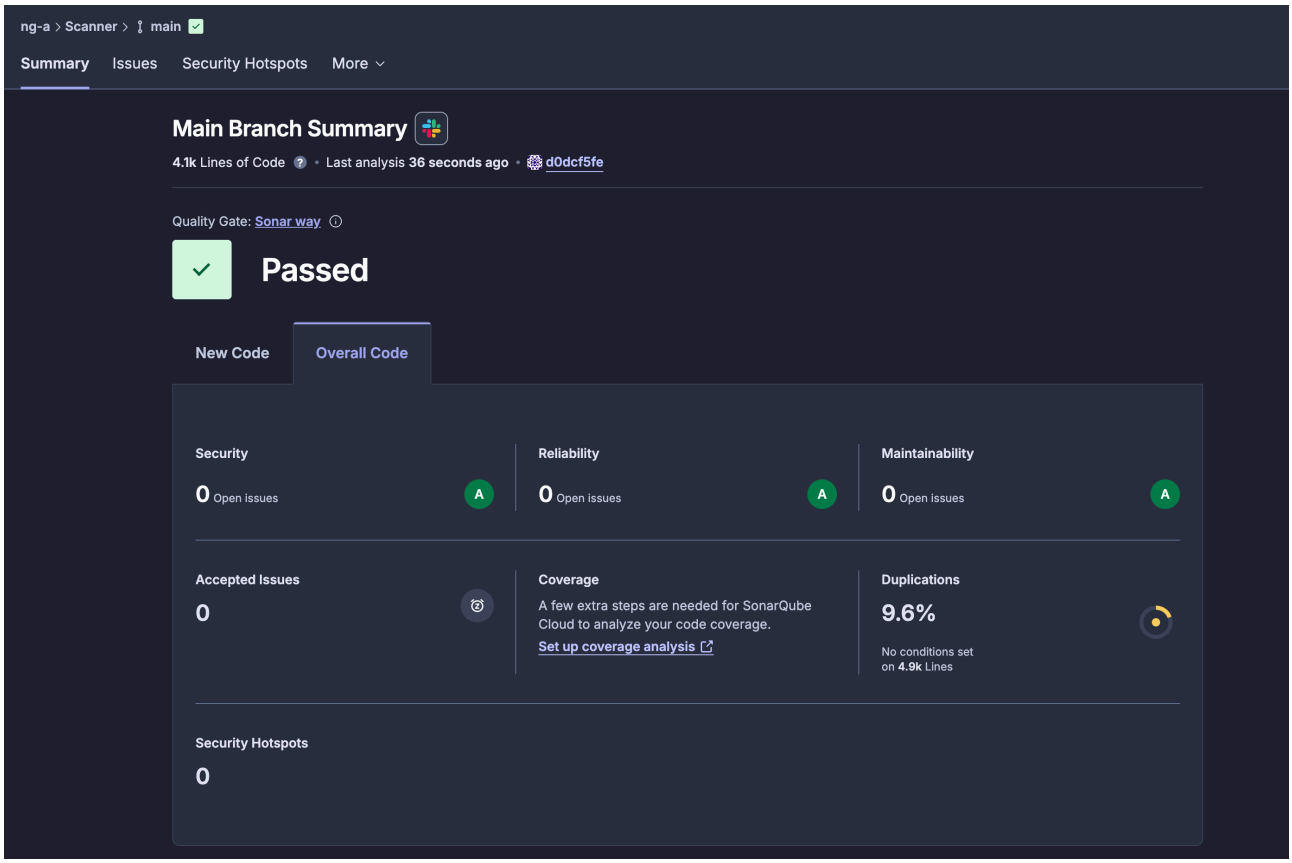


Рисунок 4.10 – Результати аналізу SonarQube

Як показано на рисунку 4.10, SonarQube проаналізував 4.1 тисячі рядків коду. Код відповідає вимогам якості. У розділі безпеки не виявлено відкритих проблем. Надійність коду та маштабованість також оцінені на найвищому рівні без відкритих проблем. Рівень дублювання коду становить 9.6%.

Результати аналізу за допомогою Semgrep та SonarQube підтверджують відсутність вразливостей та порушень стандартів кодування у програмному коді розробленого засобу аналізу вебсайтів. Це свідчить про дотримання принципів безпечного програмування та коректне використання можливостей мови Rust.

4.3 Інтеграційне тестування засобу

Інтеграційне тестування розробленого засобу аналізу безпеки вебсайтів проводилось для перевірки взаємодії всіх його компонентів у робочому середовищі. Основна увага була зосереджена на практичному тестуванні на реальних вразливих навчальних платформах Damn Vulnerable Web Application

(DVWA) [73] та Hackazon [74], які містять набір відомих, навмисно створених вразливостей.

Важливою частиною тестування була перевірка інтерфейсу командного рядка, який є основним засобом взаємодії користувача з програмним засобом. Цей інтерфейс надає гнучкі можливості для налаштування параметрів сканування, додавання заголовків автентифікації, контролю глибини та масштабу аналізу, а також управління процесами оновлення баз даних і форматами вихідних звітів. Повна довідка з використання командного рядка засобу, що демонструє всі доступні опції та їх призначення, представлена на рисунку 4.11.

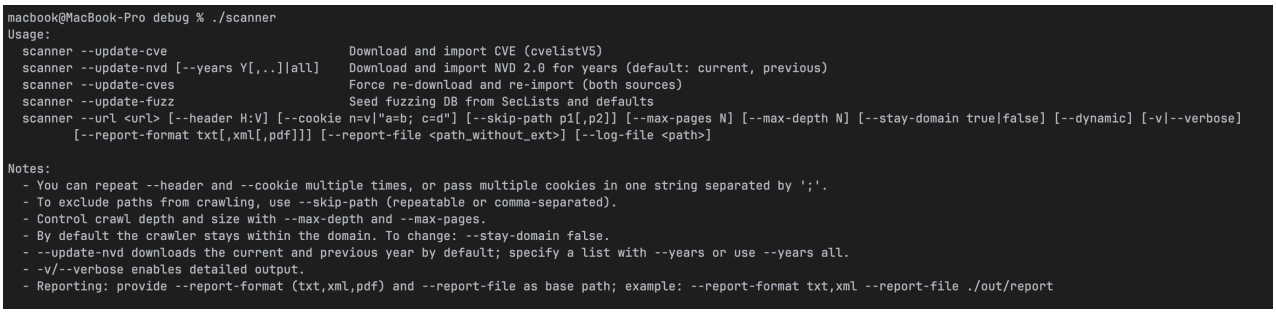


Рисунок 4.11 – Інтерфейс командного рядка

Як видно з рисунку 4.11, інтерфейс командного рядка засобу надає гнучкі можливості для налаштування процесу аналізу безпеки вебсайтів. Основними командами є:

- --update-cve – завантажує та імпортує базу даних CVE;
- --update-nvd – завантажує дані NVD за вказані роки;
- --update-cves – виконує повне оновлення всіх джерел;
- --update-fuzz – наповнює базу тестових навантажень для фазингу;
- --url <url> – запускає сканування конкретного сайту з опціями:
  - --header та --cookie – для автентифікації;
  - --skip-path – для виключення шляхів з обходу;
  - --max-pages та --max-depth – для контролю масштабу сканування;
  - --stay-domain – для визначення меж краулінга;
  - --threads – кількість потоків для сканування;

- --dynamic – для активації динамічного тестування;
- -v – для деталізованого виводу;
- --report-format – для вибору форматів звітів txt,xml,pdf;
- --report-file – для завдання імені результатів;
- --log-file – для збереження логів;

Завдяки такій структурі CLI інструмент дає можливість керувати всіма етапами роботи, починаючи від оновлення джерел інформації про вразливості і закінчуючи генерацією результатів у потрібному форматі. Користувач може самостійно визначати глибину та обсяг перевірки, обмежувати межі сканування, застосовувати автентифікацію та виключати непотрібні розділи сайту.

Процес інтеграційного тестування розпочинався саме з використання цих команд для налаштування та запуску повного циклу сканування тестових застосунків. На рисунку 4.12 представлено конкретний приклад запуску сканування DVWA з комплексом параметрів, що включають автентифікацію, обмеження на обхід та активацію динамічного тестування.

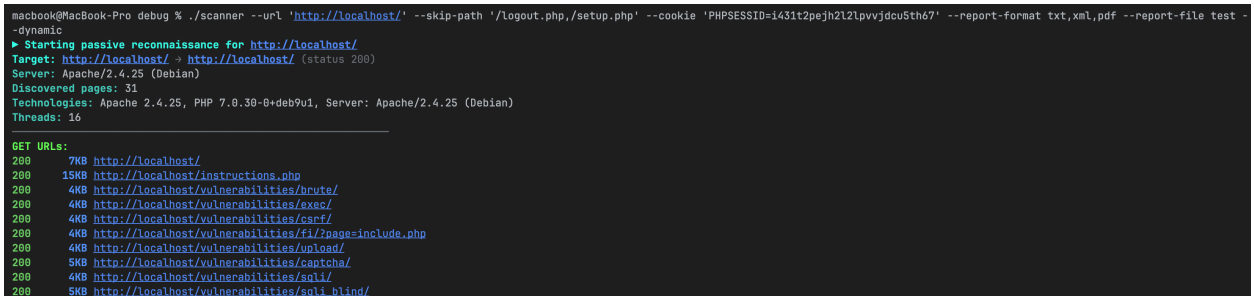


Рисунок 4.12 – Запуск сканування DVWA з параметрами

На рисунку 4.12 показано запуск сканування DVWA з параметрами, що дозволяють отримати доступ до вразливих розділів застосунку та уникнути небажаних побічних ефектів. Використання сесійних cookies для автентифікації, виключення критичних шляхів та активація динамічного сканування. Успішний запуск підтверджує коректну обробку параметрів командного рядка.

Першим етапом роботи засобу є пасивна розвідка, результати якої для тестового застосунку DVWA представлено на рисунку 4.13. Цей етап має критичне значення для подальшого аналізу, оскільки саме тут формується базова

інформаційна модель цільового вебсайту, що використовується всіма наступними модулями.

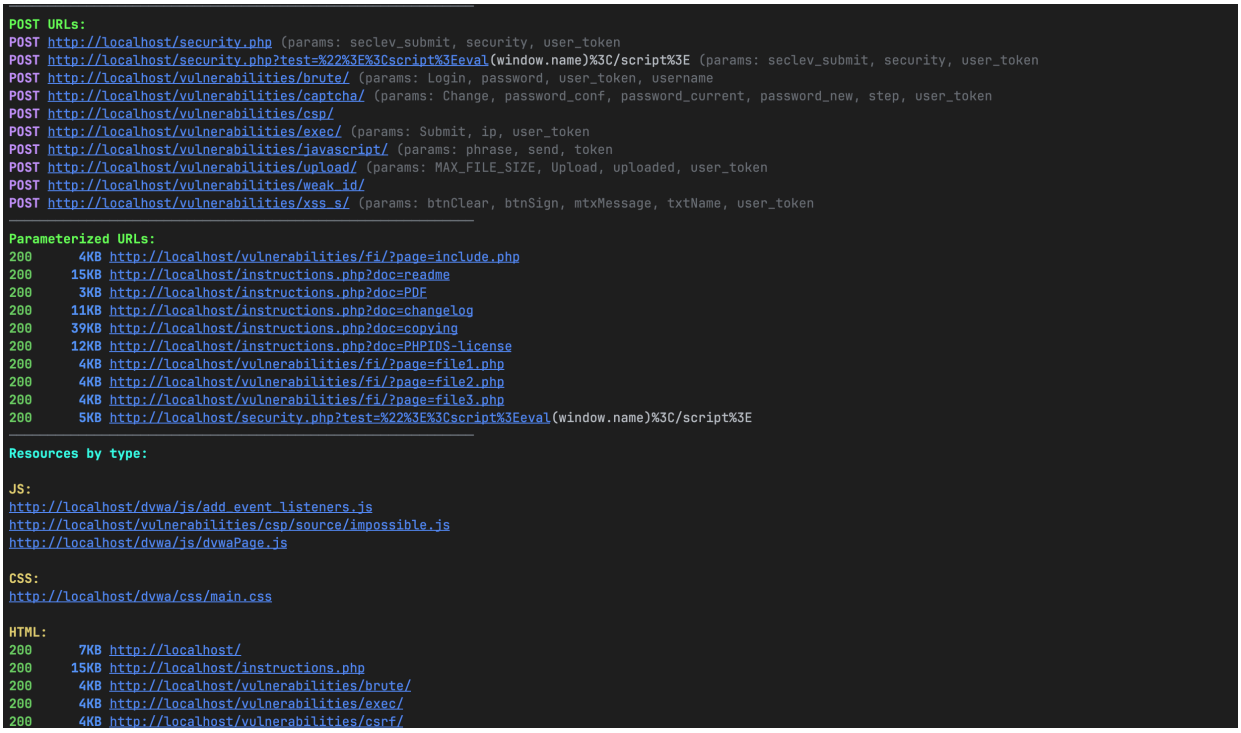


Рисунок 4.13 – Результати пасивної розвідки DVWA

Рисунок 4.13 відображає детальні результати роботи модуля пасивної розвідки. У результаті сформовано повну карту структури сайту: визначено доступні сторінки через GET-методи, ідентифіковано всі форми з їхніми параметрами для POST-запитів, а також зібрано інформацію про статичні ресурси. Ці дані стають основою для наступних етапів аналізу.

На основі технологічного стеку, виявленого під час пасивної розвідки, модуль статичного аналізу виконує пошук відомих вразливостей у локальній базі даних CVE. Результати цього аналізу для тестового застосунку представлені на рисунку 4.14, що демонструє ефективність інтеграції з актуальними джерелами інформації про загрози.

Найбільш важливим етапом тестування був динамічний аналіз через фазинг, який імітує реальні атаки на вебзастосунок шляхом відправки спеціально сформованих запитів. Результати активного сканування DVWA представлені на рисунку 4.15, що демонструє здатність засобу виявляти активні вразливості, які неможливо знайти пасивними методами.

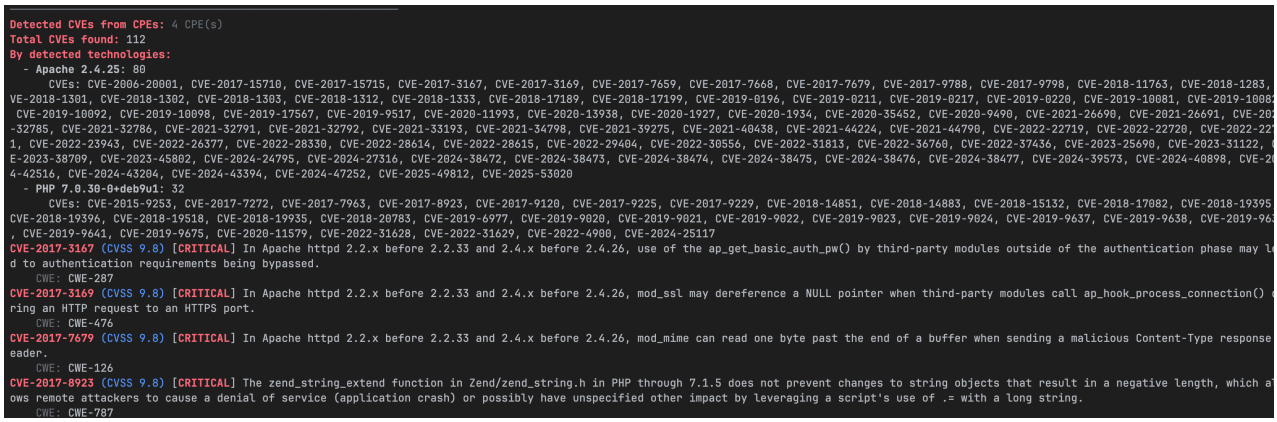


Рисунок 4.15 – Результати динамічного аналізу вразливостей DVWA

Як видно з рисунку 4.15, статичний аналіз виявив 113 відомих вразливостей. Зокрема, для вебсервера Apache версії 2.4.25 знайдено 80 потенційних загроз, а для PHP версії 7.0.30 – 32 вразливості. Серед виявлених проблем присутні критичні вразливості з максимальним рівнем CVSS 9.8, що свідчить про серйозні ризики безпеки. Модуль статичного аналізу не лише ідентифікує вразливості, але й надає детальну інформацію про кожну з них, включаючи опис проблеми, її категорію за CWE та рекомендації щодо усунення.

Найбільш важливим етапом тестування був динамічний аналіз, який імітує реальні атаки на вебзастосунок шляхом відправки спеціально сформованих запитів. Результати активного сканування DVWA представлені на рисунку 4.16, що наочно демонструє здатність засобу виявляти активні вразливості, які неможливо знайти пасивними методами.

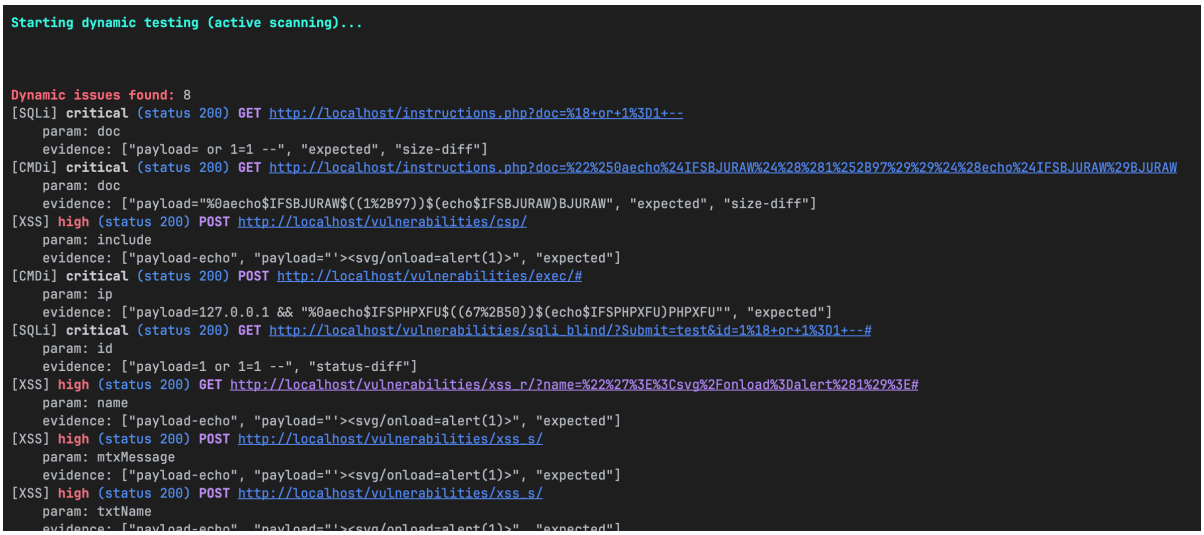
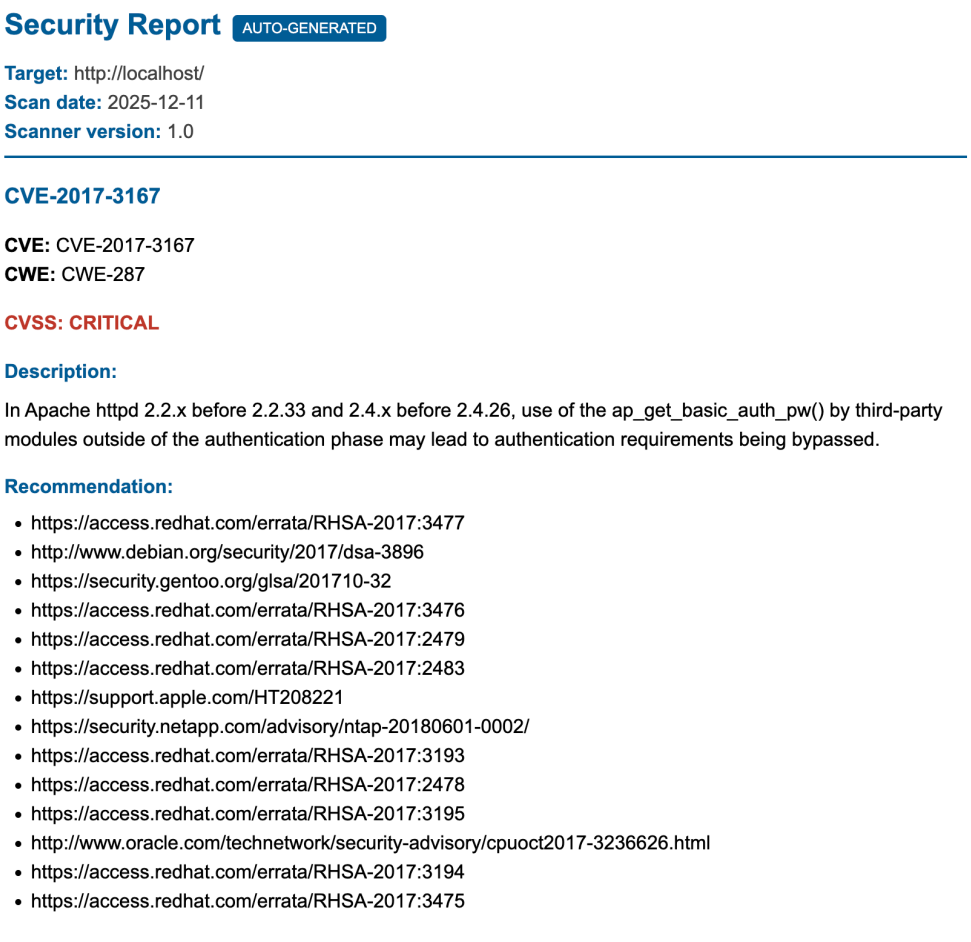


Рисунок 4.16 – Результати динамічного тестування DVWA

Рисунок 4.16 демонструє ключові результати динамічного тестування, де засіб виявив 8 активних вразливостей у DVWA різних типів та рівнів критичності. Серед них є критичні SQL-ін'єкції у параметрах doc та id, критичні вразливості виконання команд у параметрах, що призводять до можливості виконання довільних команд на сервері, та XSS у різних формах та параметрах застосунку.

Для кожної виявленої вразливості система надає детальну інформацію: тип та рівень критичності, конкретний URL та параметр, де виявлено проблему, використані тестові навантаження та конкретні докази виявлення. Така деталізація дозволяє не лише констатувати факт наявності вразливості, але й надати конкретні докази для її відтворення та подальшого усунення. Фінальним етапом стала успішна генерація звітів у всіх зазначених форматах. Фрагмент PDF-звіту представлено на рисунку 4.17.



Security Report AUTO-GENERATED

Target: http://localhost/
Scan date: 2025-12-11
Scanner version: 1.0

CVE-2017-3167

CVE: CVE-2017-3167
CWE: CWE-287

CVSS: CRITICAL

Description:

In Apache httpd 2.2.x before 2.2.33 and 2.4.x before 2.4.26, use of the ap_get_basic_auth_pw() by third-party modules outside of the authentication phase may lead to authentication requirements being bypassed.

Recommendation:

- https://access.redhat.com/errata/RHSA-2017:3477
- http://www.debian.org/security/2017/dsa-3896
- https://security.gentoo.org/glsa/201710-32
- https://access.redhat.com/errata/RHSA-2017:3476
- https://access.redhat.com/errata/RHSA-2017:2479
- https://access.redhat.com/errata/RHSA-2017:2483
- https://support.apple.com/HT208221
- https://security.netapp.com/advisory/ntap-20180601-0002/
- https://access.redhat.com/errata/RHSA-2017:3193
- https://access.redhat.com/errata/RHSA-2017:2478
- https://access.redhat.com/errata/RHSA-2017:3195
- http://www.oracle.com/technetwork/security-advisory/cpuoct2017-3236626.html
- https://access.redhat.com/errata/RHSA-2017:3194
- https://access.redhat.com/errata/RHSA-2017:3475

Рисунок 4.17 – Фрагмент PDF-звіту з результатами сканування DVWA

Як видно з рисунку 4.17, PDF-звіт містить стандартизовану структуру, що забезпечує зручність аналізу та подальшої роботи з результатами. Кожен запис включає унікальний ідентифікатор CVE, категорію CWE, оцінку критичності за CVSS, детальний опис проблеми та посилання на рекомендації щодо виправлення від різних постачальників. Такий формат дозволяє не тільки швидко оцінити загальний стан безпеки застосунку, але й отримати конкретні вказівки для усунення виявлених недоліків.

Продовжуючи інтеграційне тестування, було протестовано складніший навчальний застосунок Hackazon, що імітує функціонал повноцінного інтернет-магазину. Результати статичного сканування Hackazon представлені на рисунку 4.18, що демонструє здатність засобу ефективно аналізувати великі та складні вебзастосунки з численними сторінками та ресурсами.

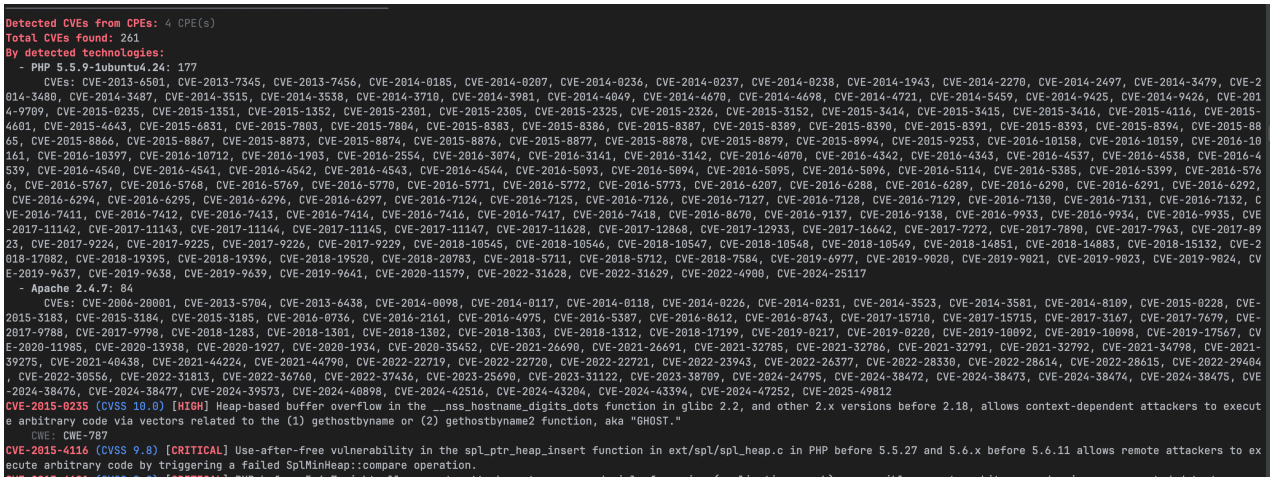


Рисунок 4.18 – Результати статичного сканування Hackazon

Як видно з рисунку 4.18, статичний аналіз виявив 261 відому вразливість у Hackazon, з них 177 пов'язані з PHP версії 5.5.9 та 84 з Apache версії 2.4.7. Серед виявлених загроз присутні критичні вразливості з високими оцінками CVSS, такі як CVE-2015-0235 (GHOST) з максимальним рівнем 10.0, що дозволяє виконувати довільний код через переповнення буфера в glibc, та численні критичні вразливості в PHP, що призводять до виконання довільного коду або відмов в обслуговуванні.

Результати динамічного тестування Hackazon представлені на рисунку 4.19, що показує здатність засобу виявляти різноманітні активні вразливості в складних застосунках.

Рисунок 4.19 – Результати динамічного сканування Hackazon

На рисунку 4.19 представлено результати динамічного тестування Hackazon, де засіб виявив 55 активних вразливостей. Серед них критичні вразливості виконання команд у різних параметрах запитів, SQL-ін'єкції у параметрі id та численні проблеми обходу шляхів (Path Traversal). Важливим аспектом є те, що сканер коректно класифікує типи вразливостей, визначає рівні критичності та надає конкретні докази виявлення, включаючи зміни статусів відповідей та розмірів контенту.

4.4 Визначення показників ефективності засобу

Для оцінки ефективності розробленого засобу аналізу безпеки було проведено порівняльне тестування з двома поширеними інструментами професійного рівня: Burp Suite Professional та OWASP ZAP. Метою порівняння було визначити здатність розробленого сканера виявляти вразливості у порівнянні з професійними аналогами та оцінити його практичну придатність для застосування в реальних умовах.

Burp Suite Professional є одним з провідних інструментів для тестування безпеки вебзастосунків. Результати сканування DVWA за допомогою Burp Suite представлені на рисунку 4.20.

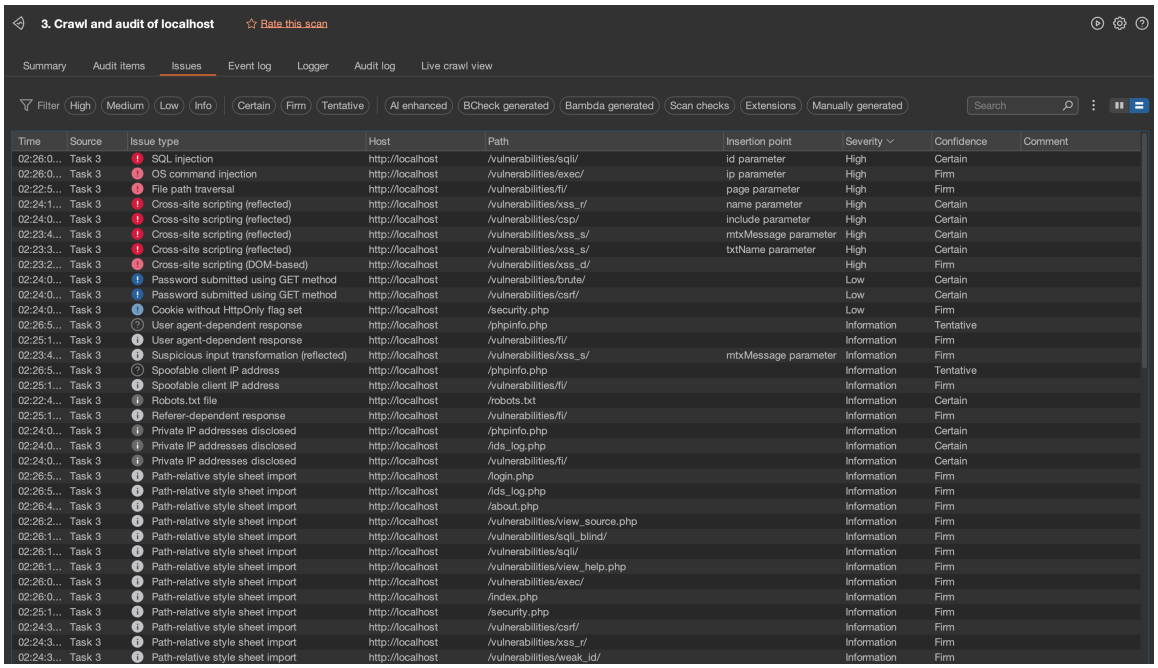


Рисунок 4.20 – Результати сканування DVWA за допомогою Burp Suite

З рисунку 4.20 видно, що Burp Suite виявив 8 вразливостей високого рівня ризику у DVWA. Серед виявлених загроз присутні ін'єкція операційних систем, SQL-ін'єкції, обхід шляхів файлової системи, відображені міжсайтові скрипти та DOM-орієнтовані XSS.

OWASP ZAP є популярним безкоштовним інструментом з відкритим кодом для тестування безпеки вебзастосунків. Як показано на рисунку 4.21.

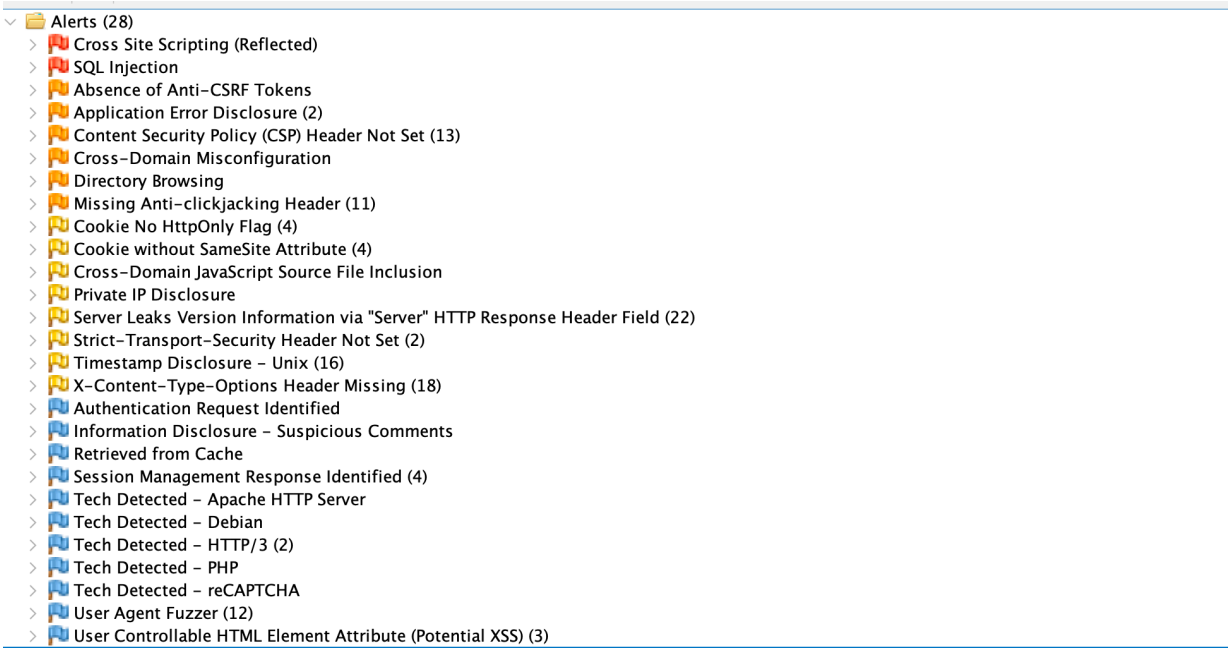


Рисунок 4.21 – Результати сканування DVWA за допомогою OWASP ZAP

З рисунку 4.21 видно, що даний інструмент виявив значно меншу кількість вразливостей у DVWA порівняно з Burp Suite. Незважаючи на високу швидкість сканування, ZAP знаходить обмежений спектр вразливостей, що знижує його ефективність для комплексного аналізу.

Розроблений засіб аналізу безпеки продемонстрував результати, порівнянні з Burp Suite. Як показано на рисунку 4.16, сканер виявив 8 активних вразливостей у DVWA, включаючи критичні SQL-ін'єкції та міжсайтові скрипти. Важливою перевагою є швидкість сканування – завдяки оптимізованій базі тестових навантажень, що містить лише актуальні та ефективні вектори атак, сканування виконується значно швидше порівняно з Burp Suite.

У таблиці 4.1 наведено порівняльну характеристику інструментів тестування безпеки.

Таблиця 4.1 – Порівняльна характеристика інструментів тестування безпеки

Критерій	**Burp Suite Professional**	**OWASP ZAP**	**Розроблений засіб**
Швидкість сканування	Повільна	Висока	Висока (фокус на актуальних загрозах, асинхронна обробка запитів)
Ефективність виявлення	Висока	Низька	Висока
Детальність звіту	Повні HTTP запити, рекомендації, класифікація	Базові описи	Опис вразливості, рекомендаціїї, критичність вразливості
Інтеграція з CVE	Обмежена (через плагіни)	Обмежена	Автоматичне оновлення
Статичний аналіз	Обмежений	Обмежений	Аналіз версій технологій
Динамічний фазинг	Розширений	Базовий	Власні бази тестів
Формати звітів	HTML, XML	HTML, XML, JSON	TXT, XML, PDF

Розроблений сканер має порівняну з Burp Suite Professional ефективність у виявленні критичних вразливостей, але працює швидше завдяки оптимізованій базі тестів. Його ключова перевага — інтеграція з базою CVE для статичного аналізу версій технологій, що дозволяє виявляти потенційні загрози. Інструмент має простий CLI інтерфейс і успішно тестувався як на навчальних, так і на

складних застосунках. Це робить його практичним рішенням для сканування вебсайтів.

4.5 Висновки з розділу

У даному розділі проведено комплексну експериментальну оцінку розробленого засобу аналізу безпеки вебсайтів. Експериментальні дослідження підтвердили коректність реалізації та працездатність розробленого засобу аналізу безпеки вебсайтів.

Модульне тестування засвідчило стабільну роботу всіх компонентів системи: модулів роботи з базами CVE/NVD, пасивної розвідки, статичного та динамічного аналізу. Статичний аналіз коду інструментами Semgrep та SonarQube не виявив вразливостей, що підтверджує високий рівень безпеки та якості розробки.

Інтеграційне тестування на платформах DVWA та Hackazon продемонструвало ефективну взаємодію модулів у повному циклі сканування. Інструмент успішно виявляє як потенційні загрози через статичний аналіз версій програмного забезпечення, так і активні вразливості, такі як SQL-ін'єкції та XSS, за допомогою динамічного фазингу, генеруючи при цьому детальні звіти.

Порівняльна оцінка з інструментами Burp Suite та OWASP ZAP показала, що розроблений засіб має порівняну з Burp Suite ефективність виявлення вразливостей, але перевершує його за швидкістю сканування завдяки оптимізованій базі тестів. Його ключовою перевагою є інтеграція з базами знань про вразливості для автоматичного статичного аналізу. Отримані результати свідчать про те, що розроблений засіб є функціонально готовим, ефективним та конкурентоспроможним рішенням для комплексного аналізу безпеки вебзастосунків.

5 ЕКОНОМІЧНА ЧАСТИНА

5.1 Проведення наукового аудиту науково-дослідної роботи

Науковий аудит даної дослідницької роботи проводиться з метою оцінки її наукового ефекту. Відповідно до методики, показник наукового ефекту визначається за формулою:

$$E_{\text{нау}} = 0{,}6 \cdot k_{\text{нов}} + 0{,}4 \cdot k_{\text{теор}} \quad (5.1)$$

де $k_{\text{нов}}$ – ступінь новизни в балах;

$k_{\text{теор}}$ – рівень теоретичного опрацювання в балах;

0,6 і 0,4 – вагові коефіцієнти.

Ступінь новизни даної роботи оцінюється як "Нова" згідно з критеріями методики, оскільки в ній запропоновано метод аналізу безпеки вебсайтів, що поєднує різнорідні бази даних вразливостей з активними методами сканування, що є вдосконаленням існуючих підходів [75]. Для категорії "Нова" діапазон оцінки становить 40–60 балів, для розрахунку приймається значення $k_{\text{нов}}$ = 55 балів.

Рівень теоретичного опрацювання відповідає характеристиці "Глибоке опрацювання проблеми", що підтверджується проведеним комплексним аналізом, розробкою математичної моделі, архітектури баз даних та алгоритмів сканування. Відповідний діапазон оцінки – 60–80 балів, для розрахунку приймається значення $k_{\text{теор}} = 75$ балів.

Підставивши значення у формулу, отримано результат:

$$E_{\text{нау}} = 0{,}6 \cdot 55 \; + \; 0{,}4 \cdot 75 = \; 33 \; + \; 30 \; = \; 63 \text{ бали}$$

Відповідно до градації, отримане значення 63 бали відповідає середньому рівню наукового ефекту. Такий результат досягнуто завдяки розробці нового методу аналізу безпеки, що удосконалює існуючі підходи через інтеграцію

різних джерел вразливостей та методик сканування, а також теоретичному обґрунтуванню всіх компонентів системи.

5.2 Проведення комерційного та технологічного аудиту науковотехнічної розробки

Метою проведення комерційного та технологічного аудиту є оцінювання науково-технічного рівня та рівня комерційного потенціалу розробленого програмного засобу для комплексного аналізу безпеки вебсайтів.

Оцінювання проводилося за методикою, що передбачає аналіз розробки за 12 критеріями. Кожному критерію відповідає п’ятибальна шкала (0–4), що детально описує ступінь відповідності продукту певній характеристиці. Шкала оцінювання наведена в таблиці 5.1.

Таблиця 5.1 – Рекомендовані критерії оцінювання науково-технічного рівня і комерційного потенціалу розробки та бальна оцінка

Бали (за 5-ти бальною шкалою)					
	0	1	2	3	4
Технічна здійсненність концепції					
1	Достовірність концепції не підтверджена	Концепція підтверджена експертними висновками	Концепція підтверджена розрахунками	Концепція перевірена на практиці	Перевірено працездатність продукту в реальних умовах
Ринкові переваги (недоліки)					
2	Багато аналогів на малому ринку	Мало аналогів на малому ринку	Кілька аналогів на великому ринку	Один аналог на великому ринку	Продукт не має аналогів на великому ринку
3	Ціна продукту значно вища за ціни аналогів	Ціна продукту дещо вища за ціни аналогів	Ціна продукту приблизно дорівнює цінам аналогів	Ціна продукту дещо нижче за ціни аналогів	Ціна продукту значно нижче за ціни аналогів
4	Технічні та споживчі властивості продукту значно гірші, ніж в	Технічні та споживчі влас-тивості продукту трохи гірші, ніж в аналогів	Технічні та споживчі властивості продукту на рівні аналогів	Технічні та споживчі властивості продукту трохи кращі, ніж в	Технічні та споживчі властивості продукту значно кращі, ніж в
5	Експлуатаційні витрати значно вищі, ніж в аналогів	Експлуатаційні витрати дещо вищі, ніж в аналогів	Експлуатаційні витрати на рівні експлуатаційних витрат аналогів	Експлуатаційні витрати трохи нижчі, ніж в аналогів	Експлуатаційні витрати значно нижчі, ніж в аналогів

Продовження таблиці 5.1

Ринкові перспективи					
6	Ринок малий і не має позитивної динаміки	Ринок малий, але має позитивну динаміку	Середній ринок з позитивною динамікою	Великий стабільний ринок	Великий ринок з позитивною динамікою
7	Активна конкуренція великих компаній на ринку	Активна конкуренція	Помірна конкуренція	Незначна конкуренція	Конкурентів немає
Практична здійсненність					
8	Відсутні фахівці як з технічної, так і з комерційної реалізації ідеї	Необхідно наймати фахівців або витрачати значні кошти та час на навчання наявних фахівців	Необхідне незначне навчання фахівців та збільшення їх штату	Необхідне незначне навчання фахівців	Є фахівці з питань як з технічної, так і з комерційної реалізації ідеї
9	Потрібні значні фінансові ресурси, які відсутні. Джерела фінансування ідеї відсутні	Потрібні незначні фінансові ресурси. Джерела фінансування відсутні	Потрібні значні фінансові ресурси. Джерела фінансування є	Потрібні незначні фінансові ресурси. Джерела фінансування є	Не потребує додаткового фінансування
10	Необхідна розробка нових матеріалів	Потрібні матеріали, що використовуються у військово промисловому комплексі	Потрібні дорогі матеріали	Потрібні досяжні та дешеві матеріали	Всі матеріали для реалізації ідеї відомі та давно використовуються у виробництві
11	Термін реалізації ідеї більший за 10 років	Термін реалізації ідеї більший за 5 років. Термін окупності інвестицій більше 10-ти років	Термін реалізації ідеї від 3-х до 5-ти років. Термін окупності інвестицій більше 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій від 3-х до 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій менше 3-х років
12	Необхідна розробка регламентних документів та отримання великої кількості дозвільних документів на виробництво та реалізацію продукту	Необхідно отримання великої кількості дозвільних документів на виробництво та реалізацію продукту, що вимагає значних коштів та часу	Процедура отримання дозвільних документів для виробництва та реалізації продукту вимагає незначних коштів та часу	Необхідно тільки повідомлення відповідним органам про виробництво та реалізацію продукту	Відсутні будь- які регламентні обмеження на виробництво та реалізацію продукту

Результати оцінювання науково-технічного рівня та комерційного потенціалу розробки наведені в таблиці 5.2.

Таблиця 5.2 – Результати оцінювання науково-технічного рівня і комерційного потенціалу розробки експертами

Критерії	**Експерт 1**	**Експерт 2**	**Експерт 3**
1. Технічна здійсненність концепції	4	4	4
2. Ринкові переваги (наявність аналогів)	3	3	3
3. Ринкові переваги (ціна продукту)	4	4	4
4. Ринкові переваги (технічні властивості)	3	3	4
5. Ринкові переваги (експлуатаційні витрати)	4	4	4
6. Ринкові перспективи (розмір ринку)	4	4	4
7. Ринкові перспективи (конкуренція)	2	3	3
8. Практична здійсненність (наявність фахівців)	4	4	4
9. Практична здійсненність (наявність фінансів)	4	4	4
10. Практична здійсненність (необхідність нових матеріалів)	4	4	4
11. Практична здійсненність (термін реалізації)	4	4	4
12. Практична здійсненність (розробка документів)	4	4	4
Сума балів (СБі)	44	45	46
Середньоарифметична сума балів (СБс)	45		

Експертні оцінки з Таблиці 5.2 є високими та збіжними. Переважна більшість критеріїв отримала максимальні бали, а сумарні оцінки експертів знаходяться у вузькому діапазоні.

На основі отриманого значення середньоарифметичної суми балів визначається загальний рівень розробки згідно з класифікацією, наведеною в таблиці 5.3.

Таблиця 5.3 – Науково-технічні рівні та комерційні потенціали розробки

Середньоарифметична сума балів СБ, розрахована на основі висновків експертів	**Науково-технічний рівень та комерційний потенціал розробки**
41…48	Високий
31…40	Вище середнього
21…30	Середній
11…20	Нижче середнього
0…10	Низький

Отримане значення середньоарифметичної суми балів 45 потрапляє в діапазон від 41 до 48 балів. Відповідно до таблиці 5.3 це свідчить про високий науково-технічний рівень та високий комерційний потенціал розробки.

Такий високий рівень досягнуто за рахунок унікального поєднання методів пасивного, статичного та динамічного аналізу в єдиному автоматизованому потоці. Це поєднання забезпечує суттєве зростання ефективності проведення тестування безпеки шляхом скорочення витрат часу та зниження залежності від кваліфікації пентестера, що розширює функціональні можливості розробки порівняно з аналогічними рішеннями на ринку.

5.3 Розрахунок витрат на здійснення науково-дослідної роботи

Витрати, пов’язані з проведенням науково-дослідної роботи на тему «Метод та засіб аналізу безпеки вебсайтів», під час планування, обліку і калькулювання собівартості науково-дослідної роботи групуються за такими статтями:

- витрати на оплату праці;
- відрахування на соціальні заходи;
- матеріали;
- паливо та енергія для науково-виробничих цілей;

– витрати на службові відрядження;

– спецустаткування для наукових (експериментальних) робіт;

– програмне забезпечення для наукових (експериментальних) робіт;

– витрати на роботи, які виконують сторонні підприємства, установи і організації;

– інші витрати;

– накладні (загальновиробничі) витрати.

5.3.1 Витрати на оплату праці

Для визначення собівартості розробки програмного засобу аналізу безпеки вебсайтів проводиться калькуляція витрат за встановленими статтями. Розрахунок охоплює повний цикл робіт тривалістю вісім місяців.

Витрати на основну заробітну плату дослідників ($З_о$) варто обчислювати у відповідності до посадових окладів працівників, за формулою 5.2.

$$З_о = \sum_{i=1}^{k} \frac{M_{ni} \cdot t_i}{T_p}, \quad (5.2)$$

де k – кількість посад дослідників, залучених до процесу досліджень;

M_{ni} – місячний посадовий оклад конкретного дослідника, грн;

t_i – кількість днів роботи конкретного дослідника, дн.;

T_p – середня кількість робочих днів в місяці, T_p = 22 дні.

Витрати на оплату праці включають основну та додаткову заробітну плату фахівців, залучених до проекту: керівника, інженера-програміста та тестувальника. Розрахунки наведено в таблиці 5.4.

Таблиця 5.4 – Витрати на заробітну плату дослідників

Найменування посади	**Місячний оклад, грн**	**Оплата за робочий день, грн**	**Кількість днів роботи**	**Витрати на заробітну плату, грн**
Керівник проекту	40 000	1 818.18	20	36 363,64
Інженер-програміст	38 000	1 727.27	22	38 000

Продовження таблиці 5.4

Найменування посади	**Місячний оклад, грн**	**Оплата за робочий день, грн**	**Кількість днів роботи**	**Витрати на заробітну плату, грн**
Інженер-тестувальник	32 000	1 454.55	10	14 545,45
Всього				88 909,09

Основна заробітна плата робітників розраховується за формулою 5.3.

$$З_{\text{р}} = \sum_{i=1}^{n} C_i \cdot t_i, \tag{5.3}$$

де C_i – погодинна тарифна ставка робітника відповідного розряду, за виконану відповідну роботу, грн/год;

t_i – час роботи робітника при виконанні визначеної роботи, год.

Погодинну тарифну ставку робітника відповідного розряду C_i можна визначити за формулою 5.4.

$$C_i = \frac{M_M \cdot K_i \cdot K_c}{T_p \cdot t_{\text{ЗМ}}}, \tag{5.4}$$

де M_M – розмір прожиткового мінімуму працездатної особи або мінімальної місячної заробітної плати (залежно від діючого законодавства), грн – наразі ММ = 8000,00 грн;

K_i – коефіцієнт міжкваліфікаційного співвідношення для встановлення тарифної ставки робітнику відповідного розряду;

K_c – мінімальний коефіцієнт співвідношень місячних тарифних ставок робітників першого розряду з нормальними умовами праці виробничих об’єднань і підприємств до законодавчо встановленого розміру мінімальної заробітної плати.

T_p – середня кількість робочих днів в місяці, приблизно T_p = 22 дні;

$t_{\text{ЗМ}}$ – тривалість зміни, год.

Витрати на основну заробітну плату робітників, залучених для технічної підтримки дослідження, розраховані згідно з формулами (5.3) та (5.4) та занесено в таблицю 5.5.

$$C_i = \frac{8000 \cdot 1{,}24 \cdot 1{,}2}{22 \cdot 8} = 67{,}64 \text{ грн.}$$

$$З_{\text{р}} = 67{,}64 \cdot 40 = 2705{,}60 \text{ грн.}$$

Таблиця 5.5 – Величина витрат на основну заробітну плату робітників

Найменування робіт	**Тривалість роботи, год**	**Розряд роботи**	**Тарифний коефіцієнт**	**Погодинна тарифна ставка, грн/год**	**Величина оплати, грн**
Дослідження відомих засобів та нормативно-правової бази	40	4	1.24	67.64	2,705.60
Аналіз архітектури та векторів атак на вебсайти	45	5	1.36	74.18	3,338.10
Розробка методу аналізу безпеки вебсайтів	50	5	1.36	74.18	3,709.00
Проектування архітектури баз даних вразливостей	35	4	1.24	67.64	2,367.40
Розробка архітектури програмного засобу	40	4	1.24	67.64	2,705.60
Вибір технологічного стеку та обґрунтування	30	4	1.24	67.64	2,029.20
Побудова алгоритмів роботи програмного засобу	55	5	1.36	74.18	4,079.90
Розробка формату звітування	25	3	1.14	62.18	1,554.50
Програмна реалізація засобу	120	5	1.36	74.18	8,901.60
Інтеграція з зовнішніми джерелами даних	60	5	1.36	74.18	4,450.80
Оптимізація продуктивності засобу	45	4	1.24	67.64	3,043.80

Продовження таблиці 5.5

Найменування робіт	**Тривалість роботи, год**	**Розряд роботи**	**Тарифний коефіцієнт**	**Погодинна тарифна ставка, грн/год**	**Величина оплати, грн**
Модульне та інтеграційне тестування	70	4	1.24	67.64	4,734.80
Статичний аналіз безпеки коду	35	4	1.24	67.64	2,367.40
Виправлення виявлених помилок та вразливостей	50	4	1.24	67.64	3,382.00
Фінальна збірка та документація	30	3	1.14	62.18	1,865.40
Всього	725	–	–	–	51 235,10

Додаткова заробітна плата призначена для компенсації виплат за невідпрацьований час, передбачений законодавством. Вона розраховується у відсотках від сукупної основної заробітної плати залучених працівників. Додаткова заробітна плата робітників розраховується за формулою 5.5.

$$З_{дод} = (З_о + З_р) \cdot \frac{Н_{дод}}{100\%}, \quad (5.5)$$

де $Н_{дод}$ – норма нарахування додаткової заробітної плати. Для даного проекту нехай середнє значення 10%.

$$З_{дод} = (88\,909{,}09 + 51\,235{,}10) \cdot \frac{10\%}{100\%} = 14\,014{,}42$$

5.3.2 Витрати на соціальні заходи

Відрахування на соціальні заходи належать обов'язкові відрахування на загальнообов'язкове державне соціальне страхування (єдиний соціальний внесок – ЄСВ), що здійснюються відповідно до чинного законодавства для забезпечення соціального захисту працівників.

Сума відрахувань розраховується як встановлений законодавством відсоток від загального фонду оплати праці, який включає як основну, так і додаткову заробітну плату всіх залучених виконавців. Розрахунок проводиться за формулою 5.6.

$$З_н = (З_о + З_р + З_{дод}) \cdot \frac{Н_{зп}}{100\%} \quad (5.6)$$

де $Н_{зп}$ – норма нарахування на заробітну плату, тобто 22%.

$$З_н = (88\,909{,}09 + 55635{,}10 + 14\,014{,}42) \cdot \frac{22\%}{100\%} = 33\,914{,}93$$

Таким чином, сума відрахувань на соціальні заходи (ЄСВ) за результатами дослідження становить 33 914,93 грн. Ця сума включається до загальних витрат на виконання науково-дослідної роботи.

5.3.3 Сировина та матеріали

Для проведення дослідження з розробки методу та засобу аналізу безпеки вебсайтів основними матеріальними витратами є носії інформації, канцелярські товари для оформлення документації, а також елементи необхідні для створення та підтримки тестового середовища проведення експериментів. Витрати на матеріали (М) у вартісному вираженні розраховуються окремо для кожного виду матеріалів за формулою 2.3.

$$M = \sum_{j-1}^{n} H_j \cdot Ц_j \cdot K_j - \sum_{j=1}^{n} B_j \cdot Ц_{вj}, \quad (5.7)$$

де H_j – норма витрат матеріалу j-го найменування, кг;

n – кількість видів матеріалів;

$Ц_j$– вартість матеріалу j-го найменування, грн/кг;

K_j– коефіцієнт транспортних витрат, (K_j= 1,1);

B_j – маса відходів j-го найменування, кг;

$Ц_{вj}$ – вартість відходів j-го найменування, грн/кг.

Оскільки основним результатом роботи є програмний засіб, витрати на сировину та матеріали є незначними. Розрахунки витрат на матеріали зведено в таблицю 5.6.

Таблиця 5.6 – Витрати на матеріали

Найменування матеріалу, марка, тип, сорт	**Ціна за 1 шт, грн**	**Норма витрат, шт**	**Величина відходів, шт**	**Ціна відходів, грн/шт**	**Коефіцієнт транспортних витрат**	**Вартість витраченого матеріалу, грн**
Накопичувач SSD (для швидкісних операцій з БД вразливостей), 500 ГБ	1800.00	1	0	0.00	1.10	1,980.00
Папір офісний А4 (пачка 500 арк.)	250.00	2	0	0.00	1.10	550.00
Картридж для лазерного принтера	600.00	1	0	0.00	1.10	660.00
Носії інформації	450.00	2	0	0.00	1.05	945.00
Канцелярський набір	300.00	1	0	0.00	1.10	330.00

Вартість спецустаткування визначається за прейскурантом гуртових цін або за даними базових підприємств за відпускними і договірними цінами. До балансової вартості устаткування окрім прейскурантної вартості входять витрати на його транспортування і монтаж, тому ці витрати беруться додатково в розмірі 10…12% від вартості устаткування.

Балансову вартість спецустаткування розраховують за формулою5.8.

$$В_{\text{прг}} = \sum_{i=1}^{k} Ц_{i\text{прг}} \cdot C_{\text{прг}.i} \cdot K_j, \qquad (5.8)$$

де $Ц_{i\text{прг}}$ – ціна придбання одиниці програмного засобу цього виду, грн;

$C_{\text{прг}.i}$ – кількість одиниць програмного забезпечення відповідного найменування, які придбані для проведення досліджень, шт.;

K_j – коефіцієнт, що враховує інсталяцію, налагодження програмного засобу тощо, (Кі = 1,10);

k – кількість найменувань програмних засобів.

Отримані результати зведено в таблицю 5.7.

Таблиця 5.7 – Витрати на придбання спецустаткування по кожному виду

Найменування програмного забезпечення	**Кількість**	**Ціна за 1 шт, грн**	**Коефіцієнт транспортних/додаткових витрат**	**Вартість, грн**
Burp Suite Professional (ліцензія на 1 рік)	1	16,000.00	1.10	17,600.00
OWASP ZAP	1	0.00	1.00	0.00
RustRover IDE (індивідуальна ліцензія)	1	0.00	1.00	0.00
Docker	1	0.00	1.10	0.00
Всього				17,600.00

5.3.4 Амортизація обладнання, програмних засобів та приміщень

В спрощеному вигляді амортизаційні відрахування по кожному виду обладнання, приміщень та програмному забезпеченню тощо можуть бути розраховані з використанням прямолінійного методу амортизації за формулою 5.9.

$$А_{обл} = \frac{Ц_б}{Т_в} \cdot \frac{t_{вик}}{12}, \quad (5.9)$$

де $Ц_б$ – балансова вартість обладнання, програмних засобів, приміщень тощо, які використовувались для проведення досліджень, грн;

$t_{вик}$ – термін використання обладнання, програмних засобів, приміщень під час досліджень, місяців;

$Т_в$ – строк корисного використання обладнання, програмних засобів, приміщень тощо, років.

Результати розрахунків наведені в таблиці 5.8.

Таблиця 5.8 – Амортизаційні відрахування по кожному виду обладнання

Найменування обладнання	**Балансова вартість, грн**	**Строк корисного використання, років**	**Термін використання обладнання, місяців**	**Амортизаційні відрахування, грн**
Міні-ПК для тестового сервера	13,750.00	4	6	1,718.75
Мережевий комутатор (Switch)	2,420.00	4	6	302.50

Продовження таблиці 5.8

Найменування обладнання	**Балансова вартість, грн**	**Строк корисного використання, років**	**Термін використання обладнання, місяців**	**Амортизаційні відрахування, грн**
Резервне джерело живлення (UPS)	4,180.00	4	6	522.50
Зовнішній жорсткий диск	3,150.00	3	6	525.00
Ноутбук для розробки	70,000.00	3	6	11,666.67
Лазерний принтер	6,000.00	4	6	750.00
ПЗ Burp Suite Professional (ліцензія)	17,600.00	2	6	4,400.00
Частка вартості офісного приміщення (орендний еквівалент)	12,000.00	20	6	300.00
Всього				**20,185.42**

5.3.5 Палива та енергія для науково-виробничих цілей

Витрати на силову електроенергію ($В_е$) можна розрахувати за формулою 5.10.

$$В_e = \sum_{i=1}^{n} W_{yi} \cdot t_i \cdot Ц_e \cdot \frac{K_{вэ}}{\eta_i}, \quad (5.10)$$

де W_{yi} – встановлена потужність обладнання на певному етапі розробки, кВт;

t_i – тривалість роботи обладнання на етапі дослідження, год;

$Ц_e$ – вартість 1 кВт-години електроенергії, грн, $Ц_e$ = 10,5 грн;

$K_{вэ}$ – коефіцієнт, що враховує використання потужності, $K_{вэ} < 1$, $K_{вэ} = 0{,}38$;

η_i – коефіцієнт корисної дії обладнання, $\eta_i < 1$, $\eta_i = 0{,}9$.

Проведені розрахунки зведено до таблиці 5.9.

Таблиця 5.9 – Витрати на електроенергію

Найменування обладнання	**Встановлена потужність, кВт**	**Тривалість роботи, год**	**Сума, грн**
Міні-ПК для тестового сервера	0.15	100	83.33
Мережевий комутатор (Switch)	0.05	100	27.78

Продовження таблиці 5.9

Найменування обладнання	**Встановлена потужність, кВт**	**Тривалість роботи, год**	**Сума, грн**
Резервне джерело живлення (UPS)	0.1	100	55.56
Зовнішній жорсткий диск	0.02	100	11.11
Ноутбук для розробки	0.1	160	88.89
Лазерний принтер	0.5	10	27.78
Всього	–	–	**294.45**

5.3.6 Службові відрядження

У службових відрядженнях немає потреби, тому кошти на них також не виділяються.

5.3.7 Витрати на роботи, які виконують сторонні підприємства, установи та організації

Заплановані роботи не передбачали залучення підрядників чи контрагентів, оскільки весь обсяг дослідження та розробки було виконано силами виконавчої команди проєкту. Тому витрати за цією статтею відсутні.

5.3.8 Інші витрати

До статті «Інші витрати» належать витрати, які не знайшли відображення у зазначених статтях витрат і можуть бути віднесені безпосередньо на собівартість досліджень за прямими ознаками. До таких витрат відносяться, наприклад, витрати на оплату банківських послуг, дрібний інструментарій, канцелярські товари, не включені в попередні статті, а також непередбачені дрібні витрати, пов'язані з організацією процесу дослідження.

Витрати за статтею «Інші витрати» розраховуються як 50…100% від суми основної заробітної плати дослідників та робітників за формулою 5.10.

$$I_{\text{в}} = (З_{\text{о}} + З_{\text{р}}) \cdot \frac{H_{\text{ів}}}{100\%}, \qquad (5.10)$$

де Нів – норма нарахування за статтею «Інші витрати», приймемо Нів = 50%.

$$I_{\text{в}} = (88\,909{,}09 + 55635{,}10) \cdot \frac{50\%}{100\%} = 70\,072{,}10.$$

5.3.9 Накладні (загальновиробничі) витрати

Витрати за статтею «Накладні (загальновиробничі) витрати» розраховуються як 100% від суми основної заробітної плати дослідників та робітників за формулою 5.11.

$$\text{В}_{\text{нзв}} = (\text{З}_{\text{о}} + \text{З}_{\text{р}}) \cdot \frac{H_{\text{зв}}}{100\%}, \tag{5.11}$$

де $H_{\text{зв}}$ – норма нарахування за статтею «Накладні (загальновиробничі) витрати».

$$\text{В}_{\text{нзв}} = (88\,909{,}09 + 55635{,}10) \cdot \frac{100\%}{100\%} = 140\,144{,}19.$$

Витрати на проведення науково-дослідної роботи розраховуються як сума всіх попередніх статей витрат за формулою 5.12.

$$\text{В}_{\text{заг}} = \text{З}_{\text{о}} + \text{З}_{\text{р}} + \text{З}_{\text{дод}} + \text{З}_{\text{н}} + \text{М} + \text{К}_{\text{в}} + \text{В}_{\text{спец}} + \text{В}_{\text{прг}} + \text{А}_{\text{обл}} + \text{В}_{\text{е}} + \text{В}_{\text{св}} + \text{В}_{\text{сп}} + \text{І}_{\text{в}} + \text{В}_{\text{нзв}}, \tag{5.12}$$

$$\text{В}_{\text{заг}} = 88\,909{,}09 + 51\,235{,}10 + 14\,014{,}42 + 33\,914{,}93 + 4\,465 + 17\,600 + 20\,185{,}42 + 294{,}45 + 70\,072{,}10 + 140\,144{,}19 = 440\,834{,}70\text{грн}$$

Загальні витрати ЗВ на завершення науково-дослідної (науково-технічної) роботи та оформлення її результатів розраховується за формулою 5.13.

$$\text{ЗВ} = \frac{\text{В}_{\text{заг}}}{\eta}, \tag{5.13}$$

де η – коефіцієнт, який характеризує етап (стадію) виконання науково-дослідної роботи. Оскільки наукова робота знаходиться на стадії розробки промислового зразка, то $\eta = 0{,}7$.

$$ЗВ = \frac{453\ 339{,}46}{0{,}7} = 629\ 763{,}86$$

5.4 Розрахунок економічної ефективності науково-технічної розробки від її впровадження безпосередньо замовником

Для оцінювання економічної доцільності вкладення коштів у комерціалізацію розробленого програмного засобу автоматизованого аналізу безпеки вебсайтів необхідно розрахувати показники економічної ефективності для потенційного інвестора.

Результати дослідження, проведені за темою «Метод та засіб аналізу безпеки вебсайтів», передбачають комерціалізацію у формі продажу ліцензій на програмне забезпечення протягом 4-х років реалізації на ринку кібербезпеки.

При розрахунку необхідно врахувати такі показники як:

ΔN – збільшення кількості клієнтів (користувачів), які придбають ліцензію на програмний засіб, в аналізовані періоди часу, від покращення його функціоналу та розпізнавання на ринку. Прогнозовані значення наведено у таблиці 5.10;

N – кількість клієнтів, які могли б використовувати аналогічні рішення у році до виходу нової розробки. Нехай N = 150 організацій;

$Ц_б$ – середня річна вартість ліцензії на аналогічний комерційний засіб аналізу безпеки (наприклад, ліцензія на сканери вразливостей) у році до впровадження нової розробки. Нехай Цо = 40 000 грн/рік;

$\pm\Delta Ц_о$ – зміни вартості ліцензії (зростання чи зниження) від впровадження переваг нової розробки (вища швидкість, краща точність, унікальні методи). Доцільно пропустити ΔЦо = +10 000 грн (додаткова вартість за покращені характеристики).

Таблиця 5.10 – Збільшення кількості клієнтів (ліцензій) в аналізовані періоди часу

Показник	1-й рік	2-й рік	3-й рік	4-й рік
Збільшення кількості споживачів, осіб	50	100	150	200

Враховуючи критично важливу роль забезпечення кібербезпеки в сучасному цифровому середовищі та постійне зростання складності та частоти веб атак, прогнозується значний попит на ефективні інструменти аудиту. Цільовою аудиторією продукту є IT-відділи великих корпорацій, фінансові установи, державні органи, а також компанії, що надають послуги з тестування на проникнення.

Можливе збільшення чистого прибутку у потенційного інвестора ΔПі для кожного із 4-х років комерціалізації розраховується за формулою 5.14.

$$\Delta П_i = (\pm \Delta Ц_o \cdot N + Ц_o \cdot \Delta N)_i \cdot \lambda \cdot \rho \cdot (1 - \frac{\vartheta}{100}), \quad (5.14)$$

де λ – коефіцієнт, який враховує сплату потенційним інвестором податку на додану вартість. У 2025 році ставка податку на додану вартість складає 20%, а коефіцієнт $\lambda = 0{,}8333$;

ρ – коефіцієнт, який враховує рентабельність інноваційного продукту). Нехай ρ =40%;

ϑ – ставка податку на прибуток, який має сплачувати потенційний інвестор, у 2025 році $\vartheta = 18\%$;

Розрахунок збільшення чистого прибутку за роками.

Збільшення чистого прибутку 1-го року:

$\Delta П_1 = (10\,000 \cdot 150 + 40\,000 \cdot 50) \cdot 0.8333 \cdot 0.35 \cdot (1 - 18/100) = (1\,500\,000 + 2\,000\,000) \cdot 0.8333 \cdot 0.35 \cdot 0.82 = 3\,500\,000 \cdot 0.8333 \cdot 0.35 \cdot 0.82 \approx 837\,083$ грн.

Збільшення чистого прибутку 2-го року:

$\Delta П_2 = (10\,000 \cdot 150 + 40\,000 \cdot 100) \cdot 0.8333 \cdot 0.35 \cdot 0.82 = (1\,500\,000 + 4\,000\,000) \cdot 0.8333 \cdot 0.35 \cdot 0.82 = 5\,500\,000 \cdot 0.8333 \cdot 0.35 \cdot 0.82 \approx 1\,315\,416$ грн.

Збільшення чистого прибутку 3-го року:

$\Delta П_3 = (10\,000 \cdot 150 + 40\,000 \cdot 150) \cdot 0.8333 \cdot 0.35 \cdot 0.82 = (1\,500\,000 + 6\,000\,000) \cdot 0.8333 \cdot 0.35 \cdot 0.82 = 7\,500\,000 \cdot 0.8333 \cdot 0.35 \cdot 0.82 \approx 1\,793\,750$ грн.

Збільшення чистого прибутку 4-го року:

$\Delta П_4 = (10\,000 \cdot 150 + 40\,000 \cdot 200) \cdot 0.8333 \cdot 0.35 \cdot 0.82 = (1\,500\,000 + 8\,000\,000) \cdot 0.8333 \cdot 0.35 \cdot 0.82 = 9\,500\,000 \cdot 0.8333 \cdot 0.35 \cdot 0.82 \approx 2\,272\,083$ грн.

Приведена вартість збільшення всіх чистих прибутків *ПП*, що їх може отримати потенційний інвестор від можливого впровадження та комерціалізації науково-технічної розробки:

$$ПП = \sum_{i=1}^{T} \frac{\Delta П_i}{(1+\tau)^t}, \tag{5.15}$$

де $\Delta П_i$ – збільшення чистого прибутку у кожному з років, протягом яких виявляються результати впровадження науково-технічної розробки, грн;

T – період часу, протягом якого очікується отримання позитивних результатів від впровадження та комерціалізації науково-технічної розробки, роки;

τ – ставка дисконтування, за яку можна взяти щорічний прогнозований рівень інфляції в країні, τ=0,12;

t – період часу (в роках) від моменту початку впровадження науково-технічної розробки до моменту отримання потенційним інвестором додаткових чистих прибутків у цьому році.

ПП = 837 083/1.15¹ + 1 315 416/1.15² + 1 793 750/1.15³ + 2 272 083/1.15⁴

ПП = 727 898 + 994 862 + 1 179 554 + 1 299 142 ≈ 4 201 456 грн.

Далі потрібно розрахувати величину початкових інвестицій PV, які замовник має вкласти для здійснення науково-технічної розробки. Для цього можна використати формулу 5.16.

$$PV = k_{інв} \cdot ЗВ, \tag{5.16}$$

де $k_{інв}$ – коефіцієнт, що враховує витрати інвестора на впровадження науково-технічної розробки та її комерціалізацію, прийнято $k_{інв}$=1,5;

$ЗВ$ – загальні витрати на проведення науково-технічної розробки та оформлення її результатів, що становить 629 763.86 грн.

$$PV = k_{інв} \cdot ЗВ = 1.5 \cdot 629\,763.86 = 944\,645.79 \text{ грн.}$$

Абсолютний економічний ефект $E_{абс}$ для потенційного інвестора від можливого впровадження та комерціалізації науково-технічної розробки розраховується за формулою 5.17.

$$E_{абс} = ПП - PV \tag{5.17}$$

де $ПП$ – приведена вартість зростання всіх чистих прибутків від можливого впровадження та комерціалізації науково-технічної розробки;

PV – теперішня вартість початкових інвестицій.

$$E_{абс} = ПП - PV = 4\,201\,456 - 944\,646 = 3\,256\,810 \text{ грн.}$$

Оскільки величина $E_{абс}$ має велике позитивне значення (понад 3.2 млн грн), це свідчить про високу потенційну зацікавленість інвесторів у комерціалізації даної розробки.

Внутрішня економічна дохідність інвестицій $E_в$, які можуть бути вкладені замовником у впровадження науково-технічної розробки, розраховується за формулою 5.18.

$$E_в = \sqrt[Т_ж]{1 + \frac{E_{абс}}{PV}} - 1, \tag{5.18}$$

де $E_{абс}$ – абсолютний економічний ефект вкладених інвестицій;

PV – теперішня вартість початкових інвестицій;

$T_{ж}$ – життєвий цикл науково-технічної розробки, тобто час від початку її розробки до закінчення отримування позитивних результатів від її впровадження, 4 роки.

$$E_{\text{в}} = \sqrt[1]{1 + \frac{3\ 256\ 810}{944\ 645.79}} - 1 = 0{,}454$$

Розрахувати мінімальну внутрішню економічну дохідність вкладених інвестицій $\tau_{мін}$ можна за формулою 5.19.

$$\tau_{мін} = d + f, \tag{5.19}$$

де d – середньозважена ставка за депозитними операціями в комерційних банках; в 2025 році в Україні d =0,12;

f – показник, що характеризує ризикованість вкладення інвестицій, взятий за 0,3.

$\tau_{мін}$ = 0,12 + 0,3 = 0,42 < 1,46 свідчить про те, що внутрішня економічна дохідність інвестицій $E_{в}$, які можуть бути вкладені потенційним інвестором у впровадження та комерціалізацію науково-технічної розробки вища мінімальної внутрішньої дохідності.

Період окупності інвестицій $T_{ок}$ які можуть бути вкладені потенційним інвестором у впровадження та комерціалізацію науково-технічної розробки:

$$T_{ок} = \frac{1}{E_{в}}, \tag{5.20}$$

де $E_{в}$ – внутрішня економічна дохідність вкладених інвестицій.

$$T_{ок} = 1 / 0{,}454 = 2{,}24 \text{ р.}$$

Таким чином, проведені розрахунки підтверджують високу економічну ефективність та комерційну привабливість науково-технічної розробки на тему «Метод та засіб аналізу безпеки вебсайтів» для потенційного інвестора.

Отримані результати свідчать про значний чистий приведений дохід, який перевищує 3 мільйони гривень, внутрішню норму дохідності на рівні приблизно 45% річних, що істотно вище розрахованої бар'єрної ставки, а також про короткий термін окупності інвестицій, що становить близько 2,2 роки.

Наведені показники обґрунтовують доцільність фінансування проекту та його подальшого виведення на ринок інформаційної безпеки.

5.5 Висновки до розділу

У економічній частині було виконано комплексну оцінку науково-дослідної роботи, що включала аудит її наукової цінності, комерційного потенціалу, розрахунок повної собівартості та аналіз інвестиційної привабливості. Науковий аудит дозволив кількісно оцінити ефект від проведених досліджень, який склав 63 бали, що відповідає середньому рівню. Даний результат обумовлений впровадженням нового комбінованого методу аналізу безпеки вебсайтів та його ґрунтовним теоретичним обґрунтуванням. Комерційний та технологічний аудит, проведений експертним шляхом, підтвердив високий науково-технічний рівень розробки та її значний комерційний потенціал, що оцінюється в 45 балів із 48 можливих. Ключовою перевагою, що визначає цей потенціал, є унікальна інтеграція різних методів аналізу в єдиному автоматизованому рішенні.

Детальна калькуляція витрат на проведення науково-дослідної роботи тривалістю вісім місяців визначила їх загальний обсяг у 440 834,70 гривні. З урахуванням необхідних витрат на доведення розробки до стадії промислового зразка, загальна сума необхідних інвестицій складає 629 763,86 гривні. Найбільшу питому вагу в структурі витрат мають витрати на оплату праці висококваліфікованих фахівців та накладні витрати.

Прогнозний розрахунок економічної ефективності для потенційного інвестора продемонстрував винятково високі показники. Чистий приведений дохід від проекту перевищує 3,25 мільйона гривень, що беззаперечно свідчить про його значну економічну вигоду. Внутрішня норма доходності інвестицій становить приблизно 45% річних, що істотно вище прийнятної для інвестора бар'єрної ставки. Критерієм низького ризику та швидкої віддачі є розрахунковий термін окупності інвестицій, який не перевищує 2,2 роки.

Отже, результати, отримані в економічній частині, об'єктивно підтверджують, що розробка «Метод та засіб аналізу безпеки вебсайтів» є не лише науково-обґрунтованою, але й економічно доцільною та інвестиційно привабливою. Має всі передумови для успішної комерціалізації на ринку, пропонуючи потенційному інвестору високорентабельне вкладення коштів із швидким періодом окупності.

ВИСНОВОК

У роботі розроблено метод та програмний засіб для автоматизованого аналізу безпеки вебсайтів, спрямований на виявлення вразливостей клієнтської та серверної частин з метою підвищення їх загального рівня захищеності.

Проведений комплексний аналіз сучасного стану проблеми виявив фрагментарність та орієнтацію на ручну роботу більшості відомих інструментів тестування, що дозволило обґрунтувати потребу у новому комплексному підході. Розроблений метод поєднує використання різнорідних баз даних відомих вразливостей з активними методами динамічного фазингу в єдиному автоматизованому потоці. Метод ґрунтується на математичному описі та реалізований у вигляді п’ятиетапного алгоритму, що охоплює ініціалізацію баз даних, пасивну розвідку, статичний аналіз, динамічне тестування та генерування звіту. Завдяки такому підходові вдалося досягти поєднання статичного та динамічного аналізу, включно з фазингом, в межах одного методу.

Для підтримки методу розроблено спеціалізовані бази даних на основі MongoDB, що забезпечують ефективне зберігання та актуальність інформації про загрози. Завдяки гнучкості документ-орієнтованих баз даних досягається покращення актуалізації їх вмісту та можливості до масштабування у випадку майбутніх змін у форматі подання звітів про вразливості у міжнародних базах. Останнє що є важливим за умов постійного оновлення цих баз. Практичною реалізацією став програмний засіб з модульною архітектурою, створений на мові Rust для гарантії високої продуктивності та безпеки. Інструмент надає гнучкий командний інтерфейс та здатність генерувати детальні звіти у кількох форматах.

Експериментальні дослідження підтвердили високий рівень функціональних можливостей та кількості виявлених вразливостей розробленого засобу. Тестування на вразливих навчальних платформах продемонструвало здатність засобу виявляти широкий спектр критичних вразливостей. Порівняльний аналіз з лідерами ринку показав порівняну ефективність виявлення при більшій швидкості роботи та додатковій перевазі у

вигляді інтегрованого статичного аналізу версій програмного забезпечення. Власна безпека коду інструменту була перевірена та підтверджена статичними аналізаторами.

Економічне обґрунтування підтверджує доцільність та привабливість розробки. Науковий аудит оцінив рівень наукового ефекту як середній, а комерційний аудит визнав розробку такою, що має високий науково-технічний рівень та значний комерційний потенціал. Прогнозні розрахунки економічної ефективності свідчать про високу прибутковість майбутнього впровадження та короткий термін окупності інвестицій.

Таким чином, в роботі успішно вирішено поставлене завдання. Розроблений метод та програмний засіб забезпечують сучасний, автоматизований та комплексний підхід до аналізу безпеки вебсайтів, що дозволяє суттєво підвищити функціональність та рівень автоматизації процесу виявлення вразливостей. Отримані результати мають практичну цінність та чітко продемонстровані перспективи комерціалізації, що робить роботу завершеною та готовою до практичного впровадження

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. *Баришев Ю. В. , Кравчук І. Ю.*. Порівняльний аналіз OWASP Top 10 s CWE Top 25. LIV Всеукраїнська науково-технічна конференція факультету інформаційних технологій та комп`ютерної інженерії. Вінниця, 2025. 3 с. URL: https://conferences.vntu.edu.ua/index.php/all-fitki/all-fitki-2025/paper/view/23890/19805 (accessed: 07.09.2025).

2. Kravchuk I., Baryshev Y. Analysis of Tools for Detecting Vulnerabilities from CWE Top 25. *Міжнародна конференція SMICS-2025 «Безпека сучасних інформаційно-комунікаційних систем»,* м. Львів, 16-18 жовтня 2025 року. 215-218с URL: https://smics.lnu.edu.ua/uk/zbirnyk/ (accessed: 20.10.2025).

3. Кримінальний кодекс України. Відомості Верховної Ради України (ВВР), 2001, № 25–26, ст.131. URL: https://zakon.rada.gov.ua/laws/show/2341-14#Text (дата звернення: 06.09.2025).

4. Bug Bounty Program. URL: https://www.hackerone.com/bug-bounty-programs (accessed: 10.09.2025).

5. RFC 9116. A File Format to Aid in Security Vulnerability Disclosure. Internet Engineering Task Force (IETF), E. Foudil, Y. Shafranovich. April 2022. URL: https://datatracker.ietf.org/doc/html/rfc9116

6. MDN Web Docs. Client–server overview. Mozilla Foundation. URL: https://developer.mozilla.org/en-US/docs/Learn/Server-side/First_steps/Client-Server_overview (accessed: 12.09.2025).

7. MDN Web Docs. HTTP methods. Mozilla Foundation. URL: https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods (accessed: 10.06.2025).

8. RFC 9110. HTTP Semantics. Internet Engineering Task Force (IETF), R. Fielding, M. Nottingham, J. Reschke. June 2022. URL: https://datatracker.ietf.org/doc/html/rfc9110 (accessed: 13.09.2025).

9. Cross-Origin Resource Sharing (CORS) URL: https://developer.mozilla.org/en-US/docs/Web/HTTP/Guides/CORS (accessed 13.09.2025)

10. W3C Recommendation. Cross-Origin Resource Sharing (CORS). World Wide Web Consortium (W3C), January 2020. URL: https://www.w3.org/TR/cors/ (accessed 16.09.2025).

11. RFC 9309. The Robots Exclusion Protocol. Internet Engineering Task Force (IETF), M. Koster, G. Illyes, H. Zeller, L. Sassman. September 2022. URL: https://datatracker.ietf.org/doc/html/rfc9309 (accessed 16.06.2025).

12. Sitemaps XML format. URL: https://www.sitemaps.org/protocol.html (accessed: 18.09.2025). Document Object Model (DOM). URL: https://developer.mozilla.org/en-US/docs/Web/API/Document_Object_Model (accessed: 18.09.2025

13. DOM-based vulnerabilities. PortSwigger Web Security Academy. URL: https://portswigger.net/web-security/dom-based (accessed: 23.09.2025).

14. Types of Cross-Site Scripting (XSS). URL: https://owasp.org/www-community/Types_of_Cross-Site_Scripting (accessed: 24.09.2025).

15. Server-side vulnerabilities. URL: https://portswigger.net/web-security/learning-paths/server-side-vulnerabilities-apprentice (accessed: 25.09.2025).

16. Technical Guide to Information Security Testing and Assessment. Karen Scarfone, Murugiah Souppaya, Amanda Cody, Angela Orebaugh. National Institute of Standards and Technology, 2008. DOI: https://doi.org/10.6028/NIST.SP.800-115 (accessed: 28.09.2025).

17. OWASP Web Security Testing Guide. URL: https://owasp.org/www-project-web-security-testing-guide/ (accessed: 02.10. 2025).

18. Acunetix. URL: https://www.acunetix.com/blog/category/docs/ (accessed: 03.10. 2025)

19. Aircrack-ng. URL: https://www.aircrack-ng.org/documentation.html (accessed: 03.10. 2025)

20. Astra Pentest. URL: https://www.getastra.com/pentesting/web-app (accessed: 14.05.2024)

21. Burp Suite. URL: https://portswigger.net/burp/pro (accessed: 04.10. 2025)

22. DirBuster. URL: https://dirbuster.com/ (accessed: 05.10. 2025)

23. Ffuf. URL: https://www.kali.org/tools/ffuf/ (accessed: 05.10. 2025)

24. Gobuster. URL: https://gobuster.org/ (accessed: 06.10.2025)

25. Hashcat. URL: https://hashcat.net/hashcat/ (accessed: 07.10.2025)

26. Hydra. URL: https://hackviser.com/tactics/tools/hydra (accessed: 10.10. 2025)

27. Intruder. URL: https://help.intruder.io/en/ (accessed: 14.10.2025)

28. John the Ripper. URL: https://www.openwall.com/john/ (accessed: 14.10.2025)

29. Kismet. URL: https://www.kismetwireless.net/docs/readme/intro/kismet/ (accessed: 14.10.2025)

30. Metasploit. URL: https://docs.metasploit.com/ (accessed: 18.10. 2025)

31. Nikto. URL: https://cirt.net/Nikto2 (accessed: 18.10. 2025)

32. Nmap. URL: https://nmap.org/book/man.html (accessed: 20.10. 2025)

33. Nuclei. URL: https://projectdiscovery.io/nuclei (accessed: 22.07. 2025)

34. OWASP ZAP. URL: https://www.zaproxy.org/docs/ (accessed: 22.07. 2025)

35. RustScan. URL: https://github.com/RustScan/RustScan (accessed: 26.07. 2025)

36. sqlmap. URL: https://sqlmap.org/ (accessed: 28.07. 2025)

37. Wireshark. URL: https://www.wireshark.org/docs/ (accessed: 04.08.2025)

38. Operating Systems for Ethical Hackers – A Platform Comparison of Kali Linux and Parrot OS. International Journal of Advanced Trends in Computer Science and Engineering, Vol. 10, 2021, p. 2226. DOI: 10.30534/ijatcse/2021/1041032021 (accessed: 10.10.2025)

39. Kali Linux. URL: https://www.kali.org/ (accessed: 14.10.2025)

40. Parrot Security OS. URL: https://www.parrotsec.org/ (accessed: 14.10.2025)

41. BlackArch Linux. URL: https://blackarch.org/ (accessed: 14.10.2025)

42. Kali Purple. URL: https://www.kali.org/kali-purple/ (accessed: 15.10.2025)

43. BackBox Linux. URL: https://www.backbox.org/ (accessed: 15.10.2025)

44. DEFT Linux. URL: https://www.deftlinux.net/ (accessed: 18.10.2025)

45. Pentoo Linux. URL: https://www.pentoo.ch/ (accessed: 18.10.2025)

46. Payment Card Industry Data Security Standard (PCI DSS), Requirements and Testing Procedures Version 4.0, March 2022. URL: https://www.commerce.uwo.ca/pdf/PCI-DSS-v4_0.pdf (accessed: 24.10.2025).

47. ISO/IEC 27001. URL: https://www.iso.org/obp/ui/en/#iso:std:iso-iec:27001:ed-3:v1:en (accessed: 27.10.2025).

48. Greenbone. URL: https://www.greenbone.net/en/ (accessed: 27.10.2025).

49. Nessus. URL: https://www.tenable.com/products/nessus (accessed: 02.11.2025)

50. Nexpose. URL: https://www.rapid7.com/products/nexpose/ (accessed: 02.11.2025)

51. OpenSCAP. URL: https://www.open-scap.org/ (accessed: 03.11.2025)

52. OpenVAS. URL: https://www.openvas.org/ (accessed: 04.11.2025)

53. Qualys VMDR. URL: https://www.qualys.com/apps/vulnerability-management-detection-response/ (accessed: 06.11.2025)

54. QualysGuard. URL: https://www.qualys.com/enterprise/qualysguard/ (accessed: 07.11.2025)

55. Rapid7 InsightVM. URL: https://www.rapid7.com/products/insightvm/ (accessed: 10.11.2025)

56. Retina Network Scanner. URL: https://www.beyondtrust.com/resources/datasheets/retina-network-scanner (accessed: 15.11.2025)

57. Tenable.io. URL: https://www.tenable.com/products/tenable-io (accessed: 15.11.2025)

58. Tripwire IP360. URL: https://www.tripwire.com/products/tripwire-ip360 (accessed: 15.11.2025)

59. Wiz. URL: https://www.wiz.io/ (accessed: 16.11.2025)

60. OWASP Top 10. URL: https://owasp.org/www-project-top-ten/ (accessed: 17.11.2025)

61. OWASP Top 10 - 2021. : URL https://owasp.org/Top10/ (accessed: 18.11.2025)

62. CVE (MITRE). URL: https://cve.mitre.org/ (accessed: 19.11.2025)

63. NVD (National Vulnerability Database). URL: https://nvd.nist.gov/ (accessed: 19.11.2025)

64. OSV (Open Source Vulnerabilities). URL: https://osv.dev/ (accessed: 19.11.2025)

65. Exploit DB. URL: https://www.exploit-db.com/ (accessed: 19.11.2025)

66. Vulners. URL: https://vulners.com/ (accessed: 20.11.2025)

67. VulnDB (Risk Based Security). URL: https://vulndb.cyberriskanalytics.com/ (accessed: 22.11.2025)

68. Vulnerability Metrics. URL: https://nvd.nist.gov/vuln-metrics/cvss (accessed: 23.11.2025)

69. NoSQL. URL: https://www.ibm.com/think/topics/nosql-databases (accessed: 24.11.2025)

70. MongoDB. URL: https://www.mongodb.com/docs/ (accessed 26.11.2025)

71. Semgrep. URL: https://github.com/semgrep/semgrep (accessed 28.11.2025)

72. SonarQube. URL: https://docs.sonarsource.com/sonarqube-server (accessed 04.12.2025)

73. Damn Vulnerable WEB application. URL: https://github.com/digininja/DVWA (accessed 05.12.2025)

74. Hackazon. URL: https://github.com/rapid7/hackazon (accessed 07.12.2025)

75. Методичні вказівки до виконання економічної частини магістерських кваліфікаційних робіт / Уклад. : В. О. Козловський, О. Й. Лесько, В. В. Кавецький. – Вінниця : ВНТУ, 2021. – 42 с.

ДОДАТКИ

Додаток А. ПРОТОКОЛ ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ

98

Назва роботи: Метод та засіб аналізу безпеки вебсайтів

Автор роботи: Кравчук Іван Юрійович
Тип роботи: магістерська кваліфікаційна робота
Підрозділ кафедра захисту інформації ФІТКІ, група 1 БС-24м

Коефіцієнт подібності текстових запозичень, виявлених у роботі системою StrikePlagiarism **12, 27 %**

Висновок щодо перевірки кваліфікаційної роботи (відмітити потрібне)

√ Запозичення, виявлені у роботі, є законними і не містять ознак плагіату, фабрикації, фальсифікації. Роботу прийняти до захисту

☐ У роботі не виявлено ознак плагіату, фабрикації, фальсифікації, але надмірна кількість текстових запозичень та/або наявність типових розрахунків не дозволяють прийняти рішення про оригінальність та самостійність її виконання. Роботу направити на доопрацювання.

☐ У роботі виявлено ознаки плагіату та/або текстових маніпуляцій як спроб укриття плагіату, фабрикації, фальсифікації, що суперечить вимогам законодавства та нормам академічної доброчесності. Робота до захисту не приймається.

Експертна комісія:

В. о. зав. кафедри ЗІ д. т. н., проф. _____ Володимир ЛУЖЕЦЬКИЙ

Гарант освітньої програми «Безпека інформаційних і комунікаційних систем» к.т.н., доцент _____ Олеся ВОЙТОВИЧ

Особа, відповідальна за перевірку _____ Валентина КАПЛУН

З висновком експертної комісії ознайомлений(-на)

Керівник _____ Юрій БАРИШЕВ

Здобувач _____ Іван КРАВЧУК

Додаток Б. ТЕКСТ ПРОГРАМИ

main.rs

```
#![allow(warnings)]
mod download_cve;
mod import_cve;
mod fuzzing_db;
mod crawler;
mod static_analysis;
mod dynamic_analysis;
mod report;

use colored::Colorize;

#[tokio::main]
async fn main() {
    if std::env::var("NO_COLOR").is_ok() {
        colored::control::set_override(false);
    }
    let args: Vec<String> = std::env::args().collect();

    let mut force_update_cves = args.iter().any(|a| a == "--update-cves");
    let mut update_cve = force_update_cves || args.iter().any(|a| a == "--update-cve");
    let mut update_nvd = args.iter().any(|a| a == "--update-nvd");
    let mut update_fuzz = args.iter().any(|a| a == "--update-fuzz");
    if update_cve { update_nvd = true; }
    let mut crawl_url: Option<String> = None;
    let mut header_vals: Vec<(String, String)> = Vec::new();
    let mut cookie_vals: Vec<(String, String)> = Vec::new();
    let mut skip_paths: Vec<String> = Vec::new();
    let mut max_pages: Option<usize> = None;
    let mut max_depth: Option<usize> = None;
    let mut stay_domain: bool = true;
    let mut verbose: bool = false;
    let mut nvd_years_cli: Vec<i32> = Vec::new();
    let mut nvd_all_years: bool = false;
    let mut do_dynamic: bool = args.iter().any(|a| a == "--dynamic");
    let mut http_threads_cli: Option<usize> = None;
    let mut report_formats: Vec<String> = Vec::new();
    let mut report_file_base: Option<String> = None;
    let mut log_file: Option<String> = None;
    let mut i = 0usize;

    while i < args.len() {
        if args[i] == "--header" {
            if let Some(raw) = args.get(i + 1) {
                if let Some(idx) = raw.find(':') {
                    let (name, value) = raw.split_at(idx);
                    let value = value.trim_start_matches(':').trim().to_string();
                    header_vals.push((name.trim().to_string(), value));
                } else {
                    eprintln!("--header expects 'Name: Value' format, got '{}'", raw);
                }
                i += 1;
            } else {
                eprintln!("--header requires an argument: 'Name: Value'");
            }
        } else if args[i] == "--cookie" {
            if let Some(raw) = args.get(i + 1) {
                let parts: Vec<&str> = if raw.contains(';') { raw.split(';').collect() } else {
vec![raw.as_str()] };
                let mut any_parsed = false;
                for p in parts {
                    let token = p.trim();
                    if token.is_empty() { continue; }
                    if let Some(idx) = token.find('=') {
                        let (name, value) = token.split_at(idx);
                        let value = value.trim_start_matches('=').to_string();
                        cookie_vals.push((name.trim().to_string(), value));
                        any_parsed = true;
                    }
                }
                if !any_parsed {
                    eprintln!("--cookie expects 'name=value' or 'a=b; c=d' format, got '{}'", raw);
```

```
                }
                i += 1;
            } else {
                eprintln!("--cookie requires an argument: 'name=value' or 'a=b; c=d'");
            }
        } else if args[i] == "--skip-path" {
            if let Some(raw) = args.get(i + 1) {
                let parts = raw.split(',');
                for p in parts {
                    let mut t = p.trim();
                    if let Some((head, _frag)) = t.split_once('#') { t = head.trim(); }
                    if t.is_empty() { continue; }
                    let norm = if t.starts_with('/') { t.to_string() } else { format!("/{}", t) };
                    skip_paths.push(norm);
                }
                i += 1;
            } else {
                eprintln!("--skip-path requires an argument: a path prefix, e.g. /logout or
logout");
            }
        } else if args[i] == "--max-pages" {
            if let Some(raw) = args.get(i + 1) {
                if !raw.starts_with('-') {
                    match raw.parse::<usize>() {
                        Ok(v) => max_pages = Some(v),
                        Err(_) => eprintln!("--max-pages expects a positive integer, got '{}'",
raw),
                    }
                    i += 1;
                } else {
                    eprintln!("--max-pages requires a number argument");
                }
            } else {
                eprintln!("--max-pages requires a number argument");
            }
        } else if args[i] == "--max-depth" {
            if let Some(raw) = args.get(i + 1) {
                if !raw.starts_with('-') {
                    match raw.parse::<usize>() {
                        Ok(v) => max_depth = Some(v),
                        Err(_) => eprintln!("--max-depth expects a positive integer, got '{}'",
raw),
                    }
                    i += 1;
                } else {
                    eprintln!("--max-depth requires a number argument");
                }
            } else {
                eprintln!("--max-depth requires a number argument");
            }
        } else if args[i] == "--stay-domain" {
            if let Some(raw) = args.get(i + 1) {
                if !raw.starts_with('-') {
                    let val = match raw.to_ascii_lowercase().as_str() {
                        "true" | "1" | "yes" | "on" => true,
                        "false" | "0" | "no" | "off" => false,
                        _ => {
                            eprintln!("--stay-domain expects true/false, got '{}' (defaulting to
true)", raw);
                            true
                        }
                    };
                    stay_domain = val;
                    i += 1;
                } else {
                    stay_domain = true;
                }
            } else {
                stay_domain = true;
            }
        } else if args[i] == "--url" {
            if let Some(next) = args.get(i + 1) {
                if !next.starts_with('-') {
                    crawl_url = Some(next.to_string());
                    i += 1;
                } else {
                    eprintln!("--url requires a URL argument, got another flag '{}'", next);
                }
            } else {
```

```
                eprintln!("--url requires a URL argument");
            }
        } else if args[i] == "--years" {
            if let Some(raw) = args.get(i + 1) {
                if raw.trim().eq_ignore_ascii_case("all") {
                    nvd_all_years = true;
                } else {
                    for p in raw.split(',') {
                        let t = p.trim();
                        if t.is_empty() { continue; }
                        if let Ok(y) = t.parse::<i32>() { nvd_years_cli.push(y); }
                    }
                }
                i += 1;
            } else {
                eprintln!("--years requires a comma-separated list, e.g. --years 2024,2025");
            }
        } else if args[i] == "--years-all" {
            nvd_all_years = true;
        } else if args[i] == "--report-format" {
            if let Some(raw) = args.get(i + 1) {
                if !raw.starts_with('-') {
                    for f in raw.split(',') { let t = f.trim().to_ascii_lowercase(); if
!t.is_empty() { report_formats.push(t); } }
                    i += 1;
                }
            }
        } else if args[i] == "--report-file" {
            if let Some(raw) = args.get(i + 1) { if !raw.starts_with('-') { report_file_base =
Some(raw.to_string()); i += 1; } }
        } else if args[i] == "--log-file" {
            if let Some(raw) = args.get(i + 1) { if !raw.starts_with('-') { log_file =
Some(raw.to_string()); i += 1; } }
        } else if args[i] == "--threads" || args[i] == "-t" {
            if let Some(raw) = args.get(i + 1) {
                if !raw.starts_with('-') {
                    match raw.parse::<usize>() {
                        Ok(v) if v >= 1 && v <= 1024 => { http_threads_cli = Some(v); i += 1; }
                        _ => eprintln!("--threads expects an integer in range [1..1024], got '{}'",
raw),
                    }
                } else {
                    eprintln!("--threads requires a number argument");
                }
            } else {
                eprintln!("--threads requires a number argument");
            }
        } else if args[i] == "-v" || args[i] == "--verbose" {
            verbose = true;
        }
        i += 1;
    }

    if !update_cve && !update_fuzz && !update_nvd && crawl_url.is_none() {
        println!(
            "Usage:\n  scanner --update-cve                          Download and import CVE
(cvelistV5)\n  scanner --update-nvd [--years Y[,..]|all]    Download and import NVD 2.0 for years
(default: current, previous)\n  scanner --update-cves                        Force re-download and
re-import (both sources)\n  scanner --update-fuzz                        Seed fuzzing DB from
SecLists and defaults\n  scanner --url <url> [--header H:V] [--cookie n=v|\"a=b; c=d\"] [--skip-path
p1[,p2]] [--max-pages N] [--max-depth N] [--stay-domain true|false] [--dynamic] [--threads N] [-v|--
verbose]\n         [--report-format txt[,xml[,pdf]]] [--report-file <path_without_ext>] [--log-file
<path>]\n\nNotes:\n  - You can repeat --header and --cookie multiple times, or pass multiple cookies
in one string separated by ';'.\n  - To exclude paths from crawling, use --skip-path (repeatable or
comma-separated).\n  - Control crawl depth and size with --max-depth and --max-pages.\n  - By
default the crawler stays within the domain. To change: --stay-domain false.\n  - --threads N
controls the number of parallel HTTP requests during active scanning (dynamic analysis). Default:
number of available CPUs (min 4).\n  - --update-nvd downloads the current and previous year by
default; specify a list with --years or use --years all.\n  - -v/--verbose enables detailed
output.\n  - Reporting: provide --report-format (txt,xml,pdf) and --report-file as base path;
example: --report-format txt,xml --report-file ./out/report\n"
        );
        return;
    }

    if update_cve {
        if force_update_cves {
            let out_dir = download_cve::default_out_dir();
            let _ = std::fs::remove_file(out_dir.join(".extracted"));
```

```
            let _ = std::fs::remove_file(out_dir.join(".imported"));
            if let Some(inner) = download_cve::find_inner_root(&out_dir) { let _ =
std::fs::remove_file(inner.join(".imported")); }
        }

        let scan_root = match download_cve::prepare_default_data().await {
            Ok(path) => path,
            Err(err) => {
                eprintln!("Download/Extract failed: {}", err);
                std::process::exit(1);
            }
        };
        println!("Importing CVEs from {}", scan_root.display());
        if force_update_cves {
            let _ = std::fs::remove_file(scan_root.join(".imported"));
        }
        if let Err(e) = import_cve::import_all(&scan_root).await {
            eprintln!("Import error: {}", e);
            std::process::exit(1);
        }

        println!("CVE import completed.");
    }

    if update_nvd {
        let years: Vec<i32> = {
            if nvd_all_years {
                use chrono::Datelike;
                let now = chrono::Utc::now();
                let y_to = now.year();
                let y_from = 2002;
                (y_from..=y_to).collect()
            } else if !nvd_years_cli.is_empty() {
                let mut v = nvd_years_cli.clone(); v.sort(); v.dedup(); v
            } else {
                use chrono::Datelike;
                let now = chrono::Utc::now();
                let y = now.year();
                vec![y, y - 1]
            }
        };
        println!("Downloading NVD 2.0 feeds for years: {:?}", years);
        for y in years {
            if force_update_cves {
                let dir = std::path::PathBuf::from("downloads").join("nvd").join(format!("{}", y));
                let _ = std::fs::remove_file(dir.join(".imported"));
                let _ = std::fs::remove_file(dir.join(".extracted"));
            }
            match download_cve::fetch_nvd_year(y).await {
                Ok(root) => {
                    println!("Importing NVD {} from {}", y, root.display());
                    if args.iter().any(|a| a == "--update-nvd") || force_update_cves ||
nvd_all_years || !nvd_years_cli.is_empty() {
                        let _ = std::fs::remove_file(root.join(".imported"));
                    }
                    if let Err(e) = import_cve::import_all(&root).await {
                        eprintln!("NVD import error ({}): {}", y, e);
                    }
                }
                Err(err) => {
                    eprintln!("Failed to fetch NVD {}: {}", y, err);
                }
            }
        }
        println!("NVD import completed.");
    }

    if update_fuzz {
        println!("Seeding fuzzing database from SecLists (GitHub Raw or SEC_LISTS_RAW_BASE
override)...");
        if let Err(e) = fuzzing_db::seed().await {
            eprintln!("Fuzzing DB seed error: {}", e);
            std::process::exit(1);
        }
        println!("Fuzzing DB seed completed.");
    }

    async fn vuln_db_is_empty() -> bool {
        let uri = "mongodb://root:example@localhost:27017/?authSource=admin";
```

```
        match mongodb::options::ClientOptions::parse(uri).await
            .and_then(|opts| mongodb::Client::with_options(opts)) {
            Ok(client) => {
                let db = client.database("cve_db");
                let col = db.collection::<mongodb::bson::Document>("vulnerabilities");
                match col.find_one(mongodb::bson::doc!{}).await { Ok(None) => true, _ => false }
            }
            Err(_) => false,
        }
    }
    if let Some(target) = crawl_url.as_ref() {
        if !update_cve && !update_nvd && vuln_db_is_empty().await {
            println!("{}", "CVE database is empty. Performing initial download and
import...".bold().yellow());
            let scan_root = match download_cve::prepare_default_data().await {
                Ok(path) => path,
                Err(err) => {
                    eprintln!("Download/Extract failed: {}", err);
                    std::process::exit(1);
                }
            };
            let _ = std::fs::remove_file(scan_root.join(".imported"));
            if let Err(e) = import_cve::import_all(&scan_root).await {
                eprintln!("Import error: {}", e);
                std::process::exit(1);
            }
            use chrono::Datelike;
            let now = chrono::Utc::now();
            let y = now.year();
            for yy in [y, y - 1] {
                if let Ok(root) = download_cve::fetch_nvd_year(yy).await {
                    let _ = std::fs::remove_file(root.join(".imported"));
                    let _ = import_cve::import_all(&root).await;
                }
            }
            println!("{}", "Initial CVE import completed.".green());
        }

        println!("{} {}", "▶️ Starting passive reconnaissance for".bold().bright_cyan(),
target.bold());
        let header_vals_clone = header_vals.clone();
        let cookie_vals_clone = cookie_vals.clone();
        let skip_paths_clone = skip_paths.clone();
        let res = crawler::run_with_exclusions_and_limits(
            target,
            header_vals,
            cookie_vals,
            skip_paths,
            max_pages,
            max_depth,
            stay_domain,
        ).await;
        match res {
            Ok(report) => {
                fn line() { println!("{}", "─".repeat(60).dimmed()); }

                fn color_status(code: u16) -> colored::ColoredString {
                    match code {
                        200..=299 => format!("{}", code).green().bold(),
                        300..=399 => format!("{}", code).yellow().bold(),
                        400..=599 => format!("{}", code).red().bold(),
                        _ => format!("{}", code).white().bold(),
                    }
                }

                fn fmt_size_colored(sz: usize) -> colored::ColoredString {
                    let s = if sz < 1024 { format!("{}B", sz) } else { format!("{}KB", (sz as f64 /
1024.0).round() as usize) };
                    s.blue().bold()
                }

                println!(
                    "{} {} {} {} {}",
                    "Target:".bold().bright_cyan(),
                    report.target.input.bold(),
                    "→".dimmed(),
                    report.target.final_url.to_string().bold(),
                    format!("(status {})", report.target.status).dimmed()
                );
```

```
                if let Some(s) = &report.target.server { println!("{} {}", "Server:".bold().cyan(),
s); }
                if let Some(x) = &report.target.x_powered_by { println!("{} {}", "X-Powered-
By:".bold().cyan(), x); }
                if let Some(g) = &report.target.generator { println!("{} {}",
"Generator:".bold().cyan(), g); }
                println!("{} {}", "Discovered pages:".bold().cyan(), report.discovered_urls.len());
                if !report.technologies.is_empty() { println!("{} {}",
"Technologies:".bold().cyan(), report.technologies.join(", ")); }

                line();

                println!("{}", "GET URLs:".bold().bright_green());
                for p in &report.sitemap.pages {
                    println!(
                        "{:>3} {:>8} {}",
                        color_status(p.status),
                        fmt_size_colored(p.body_size),
                        p.url
                    );
                }

                use std::collections::{BTreeMap, BTreeSet};
                let mut post_forms: BTreeMap<String, BTreeSet<String>> = BTreeMap::new();
                for p in &report.sitemap.pages {
                    for f in &p.forms {
                        if f.method.to_uppercase() == "POST" {
                            let mut display = f.action.clone();
                            if let Some(pos) = display.find('#') { display.truncate(pos); }
                            let entry = post_forms.entry(display).or_default();
                            for name in &f.fields { entry.insert(name.clone()); }
                        }
                    }
                }
                if !post_forms.is_empty() {
                    line();
                    println!("{}", "POST URLs:".bold().bright_green());
                    for (url, params) in post_forms {
                        if params.is_empty() {
                            println!("{} {}", "POST".bold().magenta(), url);
                        } else {
                            let list = params.into_iter().collect::<Vec<_>>().join(", ");
                            println!("{} {}{}{}{}", "POST".bold().magenta(), url, " ".dimmed(),
"(params: ".dimmed(), format!("{}", list).dimmed());
                        }
                    }
                }

                let mut param_urls: Vec<&crate::crawler::PageData> =
report.sitemap.pages.iter().filter(|p| !p.params.is_empty()).collect();
                if !param_urls.is_empty() {
                    line();
                    println!("{}", "Parameterized URLs:".bold().bright_green());
                    for p in param_urls.drain(..) {
                        println!(
                            "{:>3} {:>8} {}",
                            color_status(p.status),
                            fmt_size_colored(p.body_size),
                            p.url
                        );
                    }
                }

                line();
                println!("{}\n", "Resources by type:".bold().bright_cyan());
                if !report.resources.scripts.is_empty() {
                    println!("{}", "JS:".bold().yellow());
                    for s in &report.resources.scripts { println!("{}", s); }
                    println!("");
                }
                if !report.resources.styles.is_empty() {
                    println!("{}", "CSS:".bold().yellow());
                    for s in &report.resources.styles { println!("{}", s); }
                    println!("");
                }
                if !report.sitemap.pages.is_empty() {
                    println!("{}", "HTML:".bold().yellow());
                    for p in &report.sitemap.pages {
                        println!(
```

```
                            "{:>3} {:>8} {}",
                            color_status(p.status),
                            fmt_size_colored(p.body_size),
                            p.url
                        );
                    }
                    println!("");
                }
                let mut other: Vec<String> = Vec::new();
                other.extend(report.resources.images.iter().cloned());
                other.extend(report.resources.third_party.iter().cloned());
                if !other.is_empty() {
                    println!("{}", "Other:".bold().yellow());
                    for o in other { println!("{}", o); }
                }

                let mut collected_static: Vec<crate::static_analysis::FoundCve> = Vec::new();
                let mut collected_dynamic: Vec<crate::dynamic_analysis::DynamicIssue> = Vec::new();

                if !report.cpes.is_empty() {
                    line();
                    println!("{} {}", "Detected CVEs from CPEs:".bold().bright_red(), format!("{}
CPE(s)", report.cpes.len()).dimmed());
                    if verbose {
                        println!("{} {}", "CPEs:".bold().red(), report.cpes.join(", ").dimmed());
                    }
                    match crate::static_analysis::analyze(&report).await {
                        Ok(list) => {
                            if list.is_empty() {
                                println!("{}", "No CVEs matched for detected CPEs.".green());
                            } else {
                                collected_static = list.clone();
                                use std::collections::HashSet;

                                let mut vp_keys: HashSet<(String, String)> = HashSet::new();
                                for s in &report.cpes {
                                    if s.starts_with(':') && s.ends_with(':') {
                                        let trimmed = &s[1..s.len()-1];
                                        let parts: Vec<&str> = trimmed.split(':').collect();
                                        if parts.len() >= 2 {
                                            vp_keys.insert((parts[0].to_lowercase(),
parts[1].to_lowercase()));
                                        }
                                    }
                                }

                                let mut per_vp: Vec<(String, usize, Vec<String>)> = Vec::new();
                                for (vend, prod) in &vp_keys {
                                    let mut cve_ids: Vec<String> = Vec::new();
                                    'outer: for it in &list {
                                        for m in &it.cpe_matches {
                                            let ml = m.to_lowercase();
                                            if ml.contains(&format!(":{}:{}:", vend, prod))
                                                || (ml.contains(vend) && ml.contains(prod)) {
                                                cve_ids.push(it.cve_id.clone());
                                                continue 'outer;
                                            }
                                        }
                                    }
                                    cve_ids.sort();
                                    cve_ids.dedup();

                                    let mut label = None::<String>;
                                    for t in &report.technologies {
                                        let tl = t.to_lowercase();
                                        if tl.contains(prod) || tl.contains(vend) {
                                            label = Some(t.clone());
                                            break;
                                        }
                                    }
                                    let name = label.unwrap_or_else(|| format!("{}:{}", vend,
prod));

                                    per_vp.push((name, cve_ids.len(), cve_ids));
                                }
                                per_vp.sort_by(|a, b| b.1.cmp(&a.1).then_with(|| a.0.cmp(&b.0)));

                                let mut union: HashSet<String> = HashSet::new();
                                for (_name, _cnt, ids) in &per_vp { for id in ids {
union.insert(id.clone()); } }
```

```
                                let total = if !per_vp.is_empty() { union.len() } else { list.len()
};
                                println!("{} {}", "Total CVEs found:".bold().bright_red(), total);
                                if !per_vp.is_empty() {
                                    println!("{}", "By detected technologies:".bold().bright_red());
                                    for (name, cnt, ids) in &per_vp {
                                        println!("  - {}: {}", name.bold(), cnt);
                                        if !ids.is_empty() {
                                            println!("      CVEs: {}", ids.join(", "));
                                        }
                                    }
                                }

                                for item in list {
                                    let sev = item.severity.clone().unwrap_or_else(||
"unknown".to_string());
                                    let sev_col = match sev.to_ascii_uppercase().as_str() {
                                        "CRITICAL" => sev.bold().bright_red(),
                                        "HIGH" => sev.bold().red(),
                                        "MEDIUM" => sev.bold().yellow(),
                                        "LOW" => sev.bold().green(),
                                        _ => sev.bold().white(),
                                    };
                                    let score = item.cvss_score.map(|v| format!("{:.1}",
v)).unwrap_or_else(|| "-".to_string());
                                    let title = item.title.unwrap_or_default();
                                    println!(
                                        "{} {} {} {}",
                                        item.cve_id.bold().bright_red(),
                                        format!("(CVSS {})", score).blue(),
                                        format!("[{}]", sev_col),
                                        title
                                    );
                                    if let Some(cwe) = &item.cwe_id { if !cwe.is_empty() {
println!("    {} {}", "CWE:".dimmed(), cwe); } }
                                    if verbose {
                                        if !item.cpe_matches.is_empty() {
                                            println!("    {} {}", "Match:".dimmed(),
item.cpe_matches.join(", "));
                                        }
                                        if !report.cpes.is_empty() {
                                            println!("    {} {}", "CPE:".dimmed(),
report.cpes.join(", "));
                                        }
                                    }
                                }
                            }
                        }
                        Err(e) => {
                            println!("{} {}", "Static analysis error:".bold().red(), e);
                        }
                    }
                }

                if do_dynamic {
                    line();
                    println!("{}", "Starting dynamic testing (active
scanning)...".bold().bright_cyan());
                    let http_threads: usize = http_threads_cli.unwrap_or_else(|| std::cmp::max(4,
std::thread::available_parallelism().map(|n| n.get()).unwrap_or(4)));
                    if verbose { println!("{} {}", "HTTP threads:".bold().cyan(), http_threads); }
                    match crate::dynamic_analysis::analyze_with_threads(&report, &header_vals_clone,
&cookie_vals_clone, &skip_paths_clone, http_threads).await {
                        Ok(list) => {
                            if list.is_empty() {
                                println!("{}", "No dynamic issues detected.".green());
                            } else {
                                collected_dynamic = list.clone();
                                use crate::crawler::VectorKind;
                                println!("{} {}", "Dynamic issues found:".bold().bright_red(),
list.len());
                                for it in &list {
                                    let kind = match it.vector {
                                        VectorKind::Xss => "XSS",
                                        VectorKind::Sqli => "SQLi",
                                        VectorKind::OpenRedirect => "OpenRedirect",
                                        VectorKind::PathTraversal => "PathTraversal",
                                        VectorKind::CmdInjection => "CMDi",
                                        VectorKind::Ssti => "SSTI",
```

```
                                    };
                                    let risk_col = match it.risk.to_ascii_lowercase().as_str() {
                                        "high" => it.risk.clone().bold().red(),
                                        "medium" => it.risk.clone().bold().yellow(),
                                        "low" => it.risk.clone().bold().green(),
                                        _ => it.risk.clone().bold().white(),
                                    };
                                    println!(
                                        "[{}] {} {} {} {}",
                                        kind,
                                        risk_col,
                                        format!("(status {})", it.status).blue(),
                                        it.method.to_uppercase().bold().magenta(),
                                        it.url
                                    );
                                    if let Some(p) = &it.parameter { println!("    param: {}", p); }
                                    println!("    evidence: {}", it.evidence);
                                }
                            }
                        }
                        Err(err) => {
                            eprintln!("Dynamic analysis error: {}", err);
                        }
                    }
                }

                if !report_formats.is_empty() {
                    let base = report_file_base.clone().unwrap_or_else(||
"scan_report".to_string());
                    let meta_map = crate::fuzzing_db::load_meta_map().await;
                    let cve_ids: Vec<String> = collected_static.iter().map(|f|
f.cve_id.clone()).collect();
                    let cve_recs = crate::static_analysis::load_cve_recommendations(&cve_ids).await;
                    let summary = crate::report::ScanSummary::from_results_with_meta(
                        &report.target.final_url.to_string(),
                        &collected_static,
                        &collected_dynamic,
                        &meta_map,
                        &cve_recs,
                    );
                    for f in &report_formats {
                        let (fmt, path) = match f.as_str() {
                            "txt" => (crate::report::ReportFormat::Txt, format!("{}.txt", base)),
                            "xml" => (crate::report::ReportFormat::Xml, format!("{}.xml", base)),
                            "html" => (crate::report::ReportFormat::Html, format!("{}.html", base)),
                            "pdf" => (crate::report::ReportFormat::Pdf, format!("{}.pdf", base)),
                            _ => continue,
                        };
                        if let Err(e) = crate::report::save_report(&summary, fmt, &path) {
                            eprintln!("Failed to save report {}: {}", path, e);
                        } else {
                            println!("Saved report: {}", path);
                        }
                    }
                    if let Some(log_path) = &log_file {
                        let txt = summary.to_txt();
                        if let Err(e) = std::fs::write(log_path, txt) {
                            eprintln!("Failed to save log file {}: {}", log_path, e);
                        } else {
                            println!("Saved log: {}", log_path);
                        }
                    }
                }
            }
            Err(err) => {
                eprintln!("Crawler error: {}", err);
                std::process::exit(1);
            }
        }
    }
}

#[cfg(test)]
mod tests {
    #[test]
    fn main_download_defaults_are_sane() {
        let url = crate::download_cve::DEFAULT_CVELIST_URL;
        assert!(url.starts_with("https://"), "DEFAULT_CVELIST_URL must start with https://");
```

```
        let out = crate::download_cve::default_out_dir();
        assert!(out.is_relative(), "default_out_dir must be a relative path");
    }
}
```

download_cve.rs

```
use std::{
    io::Cursor,
    path::{Path, PathBuf},
};

const MARKER: &str = ".extracted";

pub const DEFAULT_CVELIST_URL: &str =
    "https://github.com/CVEProject/cvelistV5/archive/refs/heads/main.zip";

const DEFAULT_OUT_DIR_STR: &str = "downloads/cvelistV5";

pub fn default_out_dir() -> PathBuf {
    PathBuf::from(DEFAULT_OUT_DIR_STR)
}
pub const NVD20_URL_TMPL: &str = "https://nvd.nist.gov/feeds/json/cve/2.0/nvdcve-2.0-
{year}.json.zip";

fn default_nvd_out_dir_for_year(year: i32) -> PathBuf {
    PathBuf::from("downloads").join("nvd").join(format!("{}", year))
}


pub async fn fetch_nvd_year(year: i32) -> Result<PathBuf, Box<dyn std::error::Error +
Send + Sync>> {
    let url = NVD20_URL_TMPL.replace("{year}", &format!("{}", year));
    let out_dir = default_nvd_out_dir_for_year(year);
    let root = fetch_and_extract(url, &out_dir).await?;
    Ok(root)
}

pub async fn fetch_and_extract<U: AsRef<str>, P: AsRef<Path>>(
    url: U,
    dest_dir: P,
) -> Result<PathBuf, Box<dyn std::error::Error + Send + Sync>> {
    let url = url.as_ref();
    let dest_dir = dest_dir.as_ref();

    tokio::fs::create_dir_all(dest_dir).await?;

    let marker_path = dest_dir.join(MARKER);
    if tokio::fs::try_exists(&marker_path).await.unwrap_or(false) {
        return Ok(dest_dir.to_path_buf());
    }

    let resp = reqwest::get(url).await?;
    if !resp.status().is_success() {
        return Err(format!("HTTP error: {} while fetching {}", resp.status(),
url).into());
    }

    let bytes = resp.bytes().await?;

    let dest = dest_dir.to_path_buf();
    let extracted_dest = tokio::task::spawn_blocking(move || {
        let cursor = Cursor::new(bytes);
        let mut zip = zip::ZipArchive::new(cursor)?;
        zip.extract(&dest)?;
        Ok::<PathBuf, Box<dyn std::error::Error + Send + Sync>>(dest)
    })
    .await??;
```

```
    let marker = dest_dir.join(MARKER);
    tokio::fs::write(marker, b"ok").await?;

    Ok(extracted_dest)
}

pub fn find_inner_root(root: &Path) -> Option<PathBuf> {
    std::fs::read_dir(root)
        .ok()?
        .filter_map(|e| e.ok())
        .map(|e| e.path())
        .find(|p| p.is_dir())
}

pub async fn prepare_default_data(
) -> Result<PathBuf, Box<dyn std::error::Error + Send + Sync>> {
    let out_dir = default_out_dir();
    println!(
        "Preparing CVE data at {}",
        out_dir.display()
    );

    let root = fetch_and_extract(DEFAULT_CVELIST_URL, &out_dir).await?;
    println!("Ready at {}", root.display());

    let scan_root = find_inner_root(&root).unwrap_or(root);
    println!("Resolved scan root: {}", scan_root.display());
    Ok(scan_root)
}

#[allow(dead_code)]
pub async fn prepare_data<U: AsRef<str>, P: AsRef<Path>>(
    url: U,
    out_dir: P,
) -> Result<PathBuf, Box<dyn std::error::Error + Send + Sync>> {
    let out_dir = out_dir.as_ref().to_path_buf();
    let root = fetch_and_extract(url, &out_dir).await?;
    Ok(find_inner_root(&root).unwrap_or(root))
}
```

fuzzind_db.rs

```
use mongodb::{
    bson::{doc, oid::ObjectId, DateTime as BsonDateTime, Document},
    options::{ClientOptions, ReturnDocument},
    Client, Collection,
};
use std::env;

fn now_bson() -> BsonDateTime { BsonDateTime::now() }

async fn load_payloads_from_seclists_or_raw(relative_paths: &[&str]) -> Vec<String> {
    const DEFAULT_RAW_BASE: &str =
"https://raw.githubusercontent.com/danielmiessler/SecLists/master/";
    let raw_base = env::var("SEC_LISTS_RAW_BASE").unwrap_or_else(|_| DEFAULT_RAW_BASE.to_string());

    let mut out: Vec<String> = Vec::new();

    for rel in relative_paths {
        let rel_path: String = if rel.starts_with('/') { (*rel).to_string() } else { format!("/{}",
rel) };
        let url = format!("{}{}", raw_base.trim_end_matches('/'), rel_path);
        match reqwest::get(&url).await {
            Ok(resp) if resp.status().is_success() => {
                match resp.text().await {
                    Ok(text) => {
                        let mut added = 0usize;
                        for line in text.lines() {
                            let l = line.trim();
                            if l.is_empty() { continue; }
                            if l.starts_with('#') { continue; }
                            out.push(l.to_string());
                            added += 1;
```

```
                        }
                        println!("Loaded {} payloads from {}", added, url);
                    }
                    Err(err) => {
                        eprintln!("Failed to read body from {}: {}", url, err);
                    }
                }
            }
            Ok(resp) => {
                eprintln!("HTTP {} while fetching {}", resp.status(), url);
            }
            Err(err) => {
                eprintln!("Request error fetching {}: {}", url, err);
            }
        }
    }

    out
}

async fn upsert_vuln(
    col: &Collection<mongodb::bson::Document>,
    title: &str,
    attack_type: &str,
    description: &str,
    recommendations: &str,
    severity: &str,
) -> mongodb::error::Result<ObjectId> {
    let filter = doc! {"title": title};
    let update = doc! {"$set": {"title": title, "attack_type": attack_type, "description":
description, "recommendations": recommendations, "severity": severity}};
    if let Some(doc) = col
        .find_one_and_update(filter.clone(), update)
        .return_document(ReturnDocument::After)
        .upsert(true)
        .await?
    {
        if let Ok(id) = doc.get_object_id("_id") { return Ok(id.clone()); }
    }

    if let Some(d) = col.find_one(filter).await? {
        if let Ok(idref) = d.get_object_id("_id") { return Ok(idref.clone()); }
    }
    Ok(ObjectId::new())
}

fn build_dataset_doc(
    target_vuln_id: ObjectId,
    category: &str,
    risk_level: &str,
    payloads: Vec<String>,
    expected_responses: Vec<String>,
    fuzz_parameters: mongodb::bson::Document,
) -> Document {
    doc! {
        "target_vuln_id": target_vuln_id,
        "category": category,
        "risk_level": risk_level,
        "payloads": payloads,
        "expected_responses": expected_responses,
        "fuzz_parameters": fuzz_parameters,
        "created_by": "seed_script",
        "created_date": now_bson(),
    }
}

fn default_payloads_xss() -> Vec<String> { vec![
    r#""'><svg/onload=alert(1)>"#.to_string(),
    r#"<script>alert(1)</script>"#.to_string(),
    r#""><img src=x onerror=alert(1)>"#.to_string(),
    r#"%3Cscript%3Ealert(1)%3C/script%3E"#.to_string(),
]}

fn default_payloads_sqli() -> Vec<String> { vec![
    r#"' OR '1'='1"#.to_string(),
    r#"' UNION SELECT NULL-- "#.to_string(),
    r#"" OR 1=1 -- "#.to_string(),
    r#"') OR ('1'='1"#.to_string(),
]}
```

```
fn default_payloads_cmdi() -> Vec<String> { vec![
    "; id".to_string(),
    "| id".to_string(),
    "&& id".to_string(),
    "$(id)".to_string(),
]}

fn default_payloads_path_traversal() -> Vec<String> { vec![
    "../../../../etc/passwd".to_string(),
    "..%2f..%2f..%2f..%2fetc%2fpasswd".to_string(),
    "..\\..\\..\\..\\windows\\win.ini".to_string(),
]}

fn default_payloads_ssti() -> Vec<String> { vec![
    "${7*7}".to_string(),
    "{{7*7}}".to_string(),
    "#{7*7}".to_string(),
    "<%= 7*7 %>".to_string(),
]}

fn default_payloads_open_redirect() -> Vec<String> { vec![
    "//evil.com".to_string(),
    "https://evil.com".to_string(),
    "\\\\evil.com/%2e%2e".to_string(),
]}

fn xss_expected() -> Vec<String> { vec!["<script", "onerror", "alert("] .into_iter().map(|s|
s.to_string()).collect() }
fn sqli_expected() -> Vec<String> { vec!["syntax error", "SQL", "mysql", "ORA-", "SQLite"]
.into_iter().map(|s| s.to_string()).collect() }
fn cmdi_expected() -> Vec<String> {
    vec![
        "uid=", "gid=", "Windows",
        "bytes from", "icmp_seq", "ttl=", "time=", "PING ",
        "Pinging ", "TTL=",
        "sh: ", "bash: ",
    ].into_iter().map(|s| s.to_string()).collect()
}
fn traversal_expected() -> Vec<String> { vec!["root:x:", "[fonts]", "[extensions]"]
.into_iter().map(|s| s.to_string()).collect() }
fn ssti_expected() -> Vec<String> { vec!["49", "Expression"].into_iter().map(|s|
s.to_string()).collect() }
fn redirect_expected() -> Vec<String> { vec!["Location:", "302 Found"].into_iter().map(|s|
s.to_string()).collect() }

fn merge_payloads(mut a: Vec<String>, b: Vec<String>) -> Vec<String> { if a.is_empty() { return b; }
a.extend(b); a.sort(); a.dedup(); a }

pub async fn seed() -> Result<(), Box<dyn std::error::Error + Send + Sync>> {
    let uri = "mongodb://root:example@localhost:27017/?authSource=admin";
    let mut client_options = ClientOptions::parse(uri).await?;
    client_options.app_name = Some("scanner-fuzzing-seeder".to_string());
    let client = Client::with_options(client_options)?;
    let db = client.database("fuzzing_db");
    let vulns_col = db.collection::<mongodb::bson::Document>("vulns");
    let data_col = db.collection::<mongodb::bson::Document>("fuzzing_datasets");

    let xss_id = upsert_vuln(&vulns_col, "Cross-Site Scripting", "XSS", "Reflected/Stored XSS via
unescaped output", "Validate and encode output, use CSP", "high").await?;
    let sqli_id = upsert_vuln(&vulns_col, "SQL Injection", "Injection", "SQL injection via
unparameterized queries", "Use parameterized queries, ORM, validate input", "critical").await?;
    let cmdi_id = upsert_vuln(&vulns_col, "Command Injection", "Injection", "Shell command injection
via unsanitized input", "Avoid shell, whitelist, escape", "critical").await?;
    let path_id = upsert_vuln(&vulns_col, "Path Traversal", "File Inclusion", "Reading arbitrary
files via traversal", "Normalize paths, deny .., use allowlists", "high").await?;
    let ssti_id = upsert_vuln(&vulns_col, "Server-Side Template Injection", "Injection", "Arbitrary
template execution", "Avoid mixing user input with templates", "critical").await?;
    let redir_id = upsert_vuln(&vulns_col, "Open Redirect", "Redirect", "Open redirect via untrusted
URL parameters", "Validate redirect destinations", "medium").await?;

    let xss_seclists = load_payloads_from_seclists_or_raw(&[
        "Fuzzing/XSS/robot-friendly/XSS-Jhaddix.txt",
        "Fuzzing/XSS/robot-friendly/XSS-RSNAKE.txt",
        "Fuzzing/XSS/human-friendly/XSS-Jhaddix.txt",
        "Fuzzing/XSS/human-friendly/XSS-RSNAKE.txt",
        "Fuzzing/XSS/human-friendly/XSS-With-Context-Jhaddix.txt",
    ]).await;
```

```
    let sqli_seclists = load_payloads_from_seclists_or_raw(&[
        "Fuzzing/Databases/SQLi/Generic-SQLi.txt",
        "Fuzzing/Databases/SQLi/quick-SQLi.txt",
        "Fuzzing/Databases/SQLi/SQLi-Polyglots.txt",
    ]).await;

    let cmdi_seclists = load_payloads_from_seclists_or_raw(&[
        "Fuzzing/command-injection-commix.txt",
    ]).await;

    let path_seclists = load_payloads_from_seclists_or_raw(&[
        "Fuzzing/LFI/LFI-LFISuite-pathtotest.txt",
        "Fuzzing/LFI/LFI-LFISuite-pathtotest-huge.txt",
        "Fuzzing/LFI/LFI-linux-and-windows_by-1N3@CrowdShield.txt",
        "Fuzzing/LFI/LFI-gracefulsecurity-linux.txt",
        "Fuzzing/LFI/LFI-gracefulsecurity-windows.txt",
    ]).await;

    let ssti_seclists = load_payloads_from_seclists_or_raw(&[
        "Fuzzing/template-engines-expression.txt",
        "Fuzzing/template-engines-special-vars.txt",
    ]).await;

    let redir_seclists = load_payloads_from_seclists_or_raw(&[
        // Redirect/navigation related
        "Fuzzing/reverse-proxy-inconsistencies.txt",
        "Discovery/Web-Content/URLs/urls-Drupal-7.20.txt",
        "Discovery/Web-Content/URLs/urls-wordpress-3.3.1.txt",
    ]).await;

    let xss_payloads = merge_payloads(xss_seclists, default_payloads_xss());
    let sqli_payloads = merge_payloads(sqli_seclists, default_payloads_sqli());
    let cmdi_payloads = merge_payloads(cmdi_seclists, default_payloads_cmdi());
    let path_payloads = merge_payloads(path_seclists, default_payloads_path_traversal());
    let ssti_payloads = merge_payloads(ssti_seclists, default_payloads_ssti());
    let redir_payloads = merge_payloads(redir_seclists, default_payloads_open_redirect());
    let mut docs: Vec<Document> = Vec::with_capacity(6);

    docs.push(build_dataset_doc(
        xss_id,
        "xss",
        "high",
        xss_payloads,
        xss_expected(),
        doc!{"methods": ["GET", "POST"], "param_styles": ["query", "form", "json"],
"injection_points": ["value"]},
    ));

    docs.push(build_dataset_doc(
        sqli_id,
        "sqli",
        "critical",
        sqli_payloads,
        sqli_expected(),
        doc!{"methods": ["GET", "POST"], "param_styles": ["query", "form", "json"],
"injection_points": ["value"]},
    ));

    docs.push(build_dataset_doc(
        cmdi_id,
        "cmdi",
        "critical",
        cmdi_payloads,
        cmdi_expected(),
        doc!{"methods": ["GET", "POST"], "param_styles": ["query", "form", "json"],
"injection_points": ["value"]},
    ));

    docs.push(build_dataset_doc(
        path_id,
        "path_traversal",
        "high",
        path_payloads,
        traversal_expected(),
        doc!{"methods": ["GET"], "param_styles": ["path", "query"], "injection_points": ["path",
"value"]},
    ));

    docs.push(build_dataset_doc(
```

```
        ssti_id,
        "ssti",
        "critical",
        ssti_payloads,
        ssti_expected(),
        doc!{"methods": ["GET", "POST"], "param_styles": ["query", "form", "json"],
"injection_points": ["value"]},
    ));

    docs.push(build_dataset_doc(
        redir_id,
        "open_redirect",
        "medium",
        redir_payloads,
        redirect_expected(),
        doc!{"methods": ["GET", "POST"], "param_styles": ["query", "form"], "injection_points":
["value"]},
    ));

    let _ = data_col.insert_many(docs).ordered(false).await?;

    Ok(())
}

#[derive(Debug, Clone)]
pub struct VulnMeta {
    pub title: String,
    pub description: String,
    pub recommendations: Vec<String>,
    pub severity: String,
}

fn split_recommendations(raw: &str) -> Vec<String> {
    let mut out: Vec<String> = Vec::new();
    for part in raw
        .replace('\r', "")
        .split(|c| c == '\n' || c == ',' || c == ';')
    {
        let t = part.trim();
        if !t.is_empty() { out.push(t.to_string()); }
    }
    if out.is_empty() { out.push(raw.trim().to_string()); }
    out
}

pub async fn load_meta_map() -> Option<std::collections::HashMap<String, VulnMeta>> {
    let uri = "mongodb://root:example@localhost:27017/?authSource=admin";
    let client = match mongodb::options::ClientOptions::parse(uri).await
        .and_then(|opts| Client::with_options(opts)) {
        Ok(c) => c,
        Err(_) => return None,
    };
    let db = client.database("fuzzing_db");
    let col = db.collection::<mongodb::bson::Document>("vulns");
    let mut cursor = match col.find(doc!{}).await { Ok(c) => c, Err(_) => return None };
    let mut map: std::collections::HashMap<String, VulnMeta> = std::collections::HashMap::new();
    use futures::StreamExt;
    while let Some(Ok(doc)) = cursor.next().await {
        let title = doc.get_str("title").unwrap_or("").to_string();
        if title.is_empty() { continue; }
        let description = doc.get_str("description").unwrap_or("").to_string();
        let severity = doc.get_str("severity").unwrap_or("").to_string();
        let rec_raw = doc.get_str("recommendations").unwrap_or("");
        let recommendations = split_recommendations(rec_raw);
        map.insert(title.to_ascii_lowercase(), VulnMeta { title, description, recommendations,
severity });
    }
    if map.is_empty() { None } else { Some(map) }
}

#[allow(dead_code)]
pub mod test_api {
    use super::*;

    pub async fn load_from_paths(relative_paths: &[&str]) -> Vec<String> {
        load_payloads_from_seclists_or_raw(relative_paths).await
    }

    pub fn merge(a: Vec<String>, b: Vec<String>) -> Vec<String> { super::merge_payloads(a, b) }
```

```
    pub fn defaults_xss() -> Vec<String> { super::default_payloads_xss() }
    pub fn defaults_sqli() -> Vec<String> { super::default_payloads_sqli() }
    pub fn defaults_cmdi() -> Vec<String> { super::default_payloads_cmdi() }
    pub fn defaults_path() -> Vec<String> { super::default_payloads_path_traversal() }
    pub fn defaults_ssti() -> Vec<String> { super::default_payloads_ssti() }
    pub fn defaults_redirect() -> Vec<String> { super::default_payloads_open_redirect() }
}
```

dynamic_analysis.rs

```
use crate::crawler::{ReconReport, VectorKind};
use reqwest::Url;
use reqwest::header::{HeaderMap, HeaderName, HeaderValue, COOKIE};
use std::collections::HashMap;
use futures::stream::{self, StreamExt};

#[derive(Debug, Clone)]
struct FuzzDataset {
    vector: VectorKind,
    payloads: Vec<String>,
    expected: Vec<String>,
    methods: Vec<String>,
    param_styles: Vec<String>,
    injection_points: Vec<String>,
    risk_level: Option<String>,
}

async fn load_fuzz_datasets_from_db() -> Option<HashMap<VectorKind, Vec<FuzzDataset>>> {
    use mongodb::{bson::Document, options::ClientOptions, Client};
    let uri = "mongodb://root:example@localhost:27017/?authSource=admin";
    let client = match ClientOptions::parse(uri).await.and_then(|opts| Client::with_options(opts)) {
        Ok(c) => c,
        Err(_) => return None,
    };
    let db = client.database("fuzzing_db");
    let col = db.collection::<Document>("fuzzing_datasets");
    let mut cursor = match col.find(Document::new()).await { Ok(c) => c, Err(_) => return None };
    use futures::stream::TryStreamExt;
    let mut map: HashMap<VectorKind, Vec<FuzzDataset>> = HashMap::new();
    while let Ok(Some(doc)) = cursor.try_next().await {
        let category = doc.get_str("category").unwrap_or("").to_ascii_lowercase();
        let vector = if category.contains("xss") { Some(VectorKind::Xss) }
        else if category.contains("sqli") { Some(VectorKind::Sqli) }
        else if category.contains("open_redirect") || category.contains("redirect") {
Some(VectorKind::OpenRedirect) }
        else if category.contains("path_traversal") || category.contains("lfi") ||
category.contains("traversal") { Some(VectorKind::PathTraversal) }
        else if category.contains("cmdi") || category.contains("command") {
Some(VectorKind::CmdInjection) }
        else if category.contains("ssti") { Some(VectorKind::Ssti) }
        else { None };
        let Some(vk) = vector else { continue };

        let mut payloads: Vec<String> = Vec::new();
        if let Ok(arr) = doc.get_array("payloads") {
            for v in arr { if let Some(s) = v.as_str() { let t = s.trim(); if !t.is_empty() {
payloads.push(t.to_string()); } } }
        }

        let mut expected: Vec<String> = Vec::new();
        if let Ok(arr) = doc.get_array("expected_responses") {
            for v in arr { if let Some(s) = v.as_str() { let t = s.trim(); if !t.is_empty() {
expected.push(t.to_string()); } } }
        }

        let mut methods: Vec<String> = Vec::new();
        let mut param_styles: Vec<String> = Vec::new();
        let mut injection_points: Vec<String> = Vec::new();
        if let Ok(fp) = doc.get_document("fuzz_parameters") {
            if let Ok(arr) = fp.get_array("methods") { for v in arr { if let Some(s) = v.as_str() {
methods.push(s.to_uppercase()); } } }
            if let Ok(arr) = fp.get_array("param_styles") { for v in arr { if let Some(s) =
v.as_str() { param_styles.push(s.to_ascii_lowercase()); } } }
            if let Ok(arr) = fp.get_array("injection_points") { for v in arr { if let Some(s) =
v.as_str() { injection_points.push(s.to_ascii_lowercase()); } } }
        }
        if methods.is_empty() { methods = vec!["GET".to_string(), "POST".to_string()]; }
```

```
        if param_styles.is_empty() { param_styles = vec!["query".to_string(), "form".to_string()]; }
        if injection_points.is_empty() { injection_points = vec!["value".to_string()]; }

        payloads.sort(); payloads.dedup();
        expected.sort(); expected.dedup();

        let risk_level = doc.get_str("risk_level").ok().map(|s| s.to_string());
        let ds = FuzzDataset { vector: vk.clone(), payloads, expected, methods, param_styles,
injection_points, risk_level };
        map.entry(vk).or_default().push(ds);
    }
    if map.is_empty() { None } else { Some(map) }
}

#[derive(Debug, Clone)]
pub struct DynamicIssue {
    pub url: String,
    pub parameter: Option<String>,
    pub vector: VectorKind,
    pub risk: String,
    pub evidence: String,
    pub status: u16,
    pub method: String,
}

fn pick_payloads_for(vector: &VectorKind) -> Vec<String> {
    match vector {
        VectorKind::Xss => crate::fuzzing_db::test_api::defaults_xss(),
        VectorKind::Sqli => crate::fuzzing_db::test_api::defaults_sqli(),
        VectorKind::OpenRedirect => crate::fuzzing_db::test_api::defaults_redirect(),
        VectorKind::PathTraversal => crate::fuzzing_db::test_api::defaults_path(),
        VectorKind::CmdInjection => crate::fuzzing_db::test_api::defaults_cmdi(),
        VectorKind::Ssti => crate::fuzzing_db::test_api::defaults_ssti(),
    }
}

fn default_dataset_for(vector: &VectorKind) -> FuzzDataset {
    let (payloads, expected, param_styles): (Vec<String>, Vec<String>, Vec<String>) = match vector {
        VectorKind::Xss => (crate::fuzzing_db::test_api::defaults_xss(), vec!["<script".into(),
"onerror".into(), "alert(".into()], vec!["query".into(), "form".into()]),
        VectorKind::Sqli => (crate::fuzzing_db::test_api::defaults_sqli(), vec!["sqlstate".into(),
"mysql".into(), "sqlite".into(), "ora-".into()], vec!["query".into(), "form".into()]),
        VectorKind::OpenRedirect => (crate::fuzzing_db::test_api::defaults_redirect(),
vec!["Location:".into(), "302".into()], vec!["query".into(), "form".into()]),
        VectorKind::PathTraversal => (crate::fuzzing_db::test_api::defaults_path(),
vec!["root:x:".into(), "[fonts]".into()], vec!["path".into(), "query".into()]),
        VectorKind::CmdInjection => (crate::fuzzing_db::test_api::defaults_cmdi(), vec![
            "uid=".into(), "gid=".into(), "Windows".into(),
            "bytes from".into(), "icmp_seq".into(), "ttl=".into(), "time=".into(), "PING ".into(),
            "Pinging ".into(), "TTL=".into(),
            "sh: ".into(), "bash: ".into(),
        ], vec!["query".into(), "form".into()]),
        VectorKind::Ssti => (crate::fuzzing_db::test_api::defaults_ssti(), vec!["49".into(),
"Expression".into()], vec!["query".into(), "form".into()]),
    };
    FuzzDataset { vector: vector.clone(), payloads, expected, methods: vec!["GET".into(),
"POST".into()], param_styles, injection_points: vec!["value".into()], risk_level: None }
}

fn pick_datasets_for(vector: &VectorKind, db_map: &Option<HashMap<VectorKind, Vec<FuzzDataset>>>) ->
Vec<FuzzDataset> {
    if let Some(map) = db_map {
        if let Some(list) = map.get(vector) {
            if !list.is_empty() { return list.clone(); }
        }
    }
    vec![default_dataset_for(vector)]
}

fn build_test_urls(base: &str, param_name: &str, payload: &str) -> Option<String> {
    if let Ok(mut url) = Url::parse(base) {
        let mut qp: Vec<(String, String)> = url
            .query_pairs()
            .map(|(k, v)| (k.to_string(), v.to_string()))
            .collect();
        let mut replaced = false;
        for (k, v) in qp.iter_mut() {
            if k == param_name {
                *v = payload.to_string();
```

```
                replaced = true;
                break;
            }
        }
        if !replaced {
            qp.push((param_name.to_string(), payload.to_string()))
        }
        url.query_pairs_mut().clear();
        for (k, v) in qp {
            url.query_pairs_mut().append_pair(&k, &v);
        }
        Some(url.to_string())
    } else {
        None
    }
}

fn build_get_with_params(base: &str, params: &[(String, String)]) -> Option<String> {
    if let Ok(mut url) = Url::parse(base) {
        url.query_pairs_mut().clear();
        for (k, v) in params {
            url.query_pairs_mut().append_pair(k, v);
        }
        Some(url.to_string())
    } else { None }
}

fn build_form_body(fields: &[String], target_param: &str, payload: &str) -> String {
    let mut ser = url::form_urlencoded::Serializer::new(String::new());
    for f in fields {
        if f == target_param {
            ser.append_pair(f, payload);
        } else {
            ser.append_pair(f, "test");
        }
    }
    if !fields.iter().any(|f| f == target_param) {
        ser.append_pair(target_param, payload);
    }
    ser.finish()
}

fn encode_form_component(value: &str) -> String {
    let mut ser = url::form_urlencoded::Serializer::new(String::new());
    ser.append_pair("x", value);
    let s = ser.finish();
    match s.split_once('=') { Some((_, v)) => v.to_string(), None => String::new() }
}

fn is_xss_noise_url(url: &str) -> bool {
    match Url::parse(url) {
        Ok(u) => {
            let p = u.path().to_ascii_lowercase();
            p.ends_with("/phpinfo.php") || p == "/phpinfo.php"
        }
        Err(_) => url.to_ascii_lowercase().contains("phpinfo.php"),
    }
}

fn build_value_variants(vector: &VectorKind, payload: &str) -> Vec<String> {
    let mut out: Vec<String> = Vec::new();
    let p = payload.to_string();
    match vector {
        VectorKind::CmdInjection => {
            let seeds = ["127.0.0.1", "8.8.8.8"];
            out.push(p.clone());
            for seed in &seeds {
                out.push(format!("{}; {}", seed, payload));
                out.push(format!("{} && {}", seed, payload));
                out.push(format!("{} | {}", seed, payload));
            }
        }
        VectorKind::Sqli => {
            out.push(p.clone());
            out.push(format!("1{}", payload));
            out.push(format!("test{}", payload));
        }
        _ => {
            out.push(p);
```

```
        }
    }
    out.sort();
    out.dedup();
    out.into_iter().take(4).collect()
}

fn classify(
    vector: &VectorKind,
    status: u16,
    body: &str,
    payload: &str,
    location: Option<&str>,
    baseline_status: u16,
    baseline_len: usize,
    expected: &[String],
    baseline_body: Option<&str>,
) -> Option<(String, String)> {
    let body_l = body.to_ascii_lowercase();
    let mut hints: Vec<String> = Vec::new();
    if (500..=599).contains(&status) { hints.push("5xx status".to_string()); }
    let sql_error_keywords = [
        "you have an error in your sql syntax",
        "sql syntax",
        "sqlstate",
        "unclosed quotation",
        "unterminated string",
        "warning: mysql",
        "mysqli_sql_exception",
        "psql: error",
        "pg_query(",
        "sqlite error",
        "sqlite exception",
        "ora-",
        "oracle error",
        "odbc sql server",
    ];
    let mut raw_echo = false;
    let mut encoded_echo = false;
    let mut escaped_echo = false;
    if !payload.is_empty() {
        let encoded = encode_form_component(payload);
        let html_escaped = payload
            .replace('&', "&amp;")
            .replace('<', "&lt;")
            .replace('>', "&gt;")
            .replace('"', "&quot;")
            .replace('\'', "&#x27;");
        if body.contains(payload) { raw_echo = true; }
        if !encoded.is_empty() && body.contains(&encoded) { encoded_echo = true; }
        if body.contains(&html_escaped) { escaped_echo = true; }
        if raw_echo { hints.push("payload-echo".to_string()); }
        hints.push(format!("payload={}", payload));
    }

    match vector {
        VectorKind::OpenRedirect => {
            if (300..=399).contains(&status) {
                if let Some(loc) = location {
                    if loc.contains(payload) {
                        hints.push("location-contains-payload".to_string());
                    }
                    if loc.starts_with("http://") || loc.starts_with("https://") {
                        hints.push("external-location".to_string());
                    }
                }
            } else {
                let bl = body.to_ascii_lowercase();
                if bl.contains("http-equiv=\"refresh\"") && bl.contains("url=") &&
bl.contains(&payload.to_ascii_lowercase()) {
                    hints.push("meta-refresh".to_string());
                }
                if bl.contains("window.location") && bl.contains(&payload.to_ascii_lowercase()) {
                    hints.push("js-redirect".to_string());
                }
            }
        }
        VectorKind::Xss => {
        }
```

```
        VectorKind::Sqli => {
            if sql_error_keywords.iter().any(|k| body_l.contains(k)) { hints.push("sql-
error".to_string()); }
        }
        _ => {
        }
    }

    if !expected.is_empty() {
        let allow_expected = match vector { VectorKind::Xss => raw_echo, _ => true };
        if allow_expected {
            for e in expected {
                let el = e.to_ascii_lowercase();
                if !el.is_empty() && body_l.contains(&el) {
                    if let Some(bb) = baseline_body {
                        let bbl = bb.to_ascii_lowercase();
                        if !bbl.contains(&el) {
                            hints.push("expected".to_string());
                            break;
                        }
                    } else {
                        hints.push("expected".to_string());
                        break;
                    }
                }
            }
        }
    }

    let body_len = body.len();
    if status != baseline_status {
        hints.push("status-diff".to_string());
    }
    if baseline_len > 0 {
        let diff = if body_len > baseline_len { body_len - baseline_len } else { baseline_len -
body_len };
        if diff >= 200 {
            let rel = (diff as f64) / (baseline_len as f64 + 1.0);
            if rel > 0.20 { hints.push("size-diff".to_string()); }
        }
    }

    if hints.is_empty() { return None; }

    let has_5xx = hints.iter().any(|h| h == "5xx status");
    let has_echo = hints.iter().any(|h| h == "payload-echo");
    let has_loc = hints.iter().any(|h| h == "location-contains-payload" || h == "external-
location");
    let has_errkw = hints.iter().any(|h| h == "sql-error");
    let has_delta = hints.iter().any(|h| h == "status-diff" || h == "size-diff");

    match vector {
        VectorKind::Xss => {
            if !raw_echo { return None; }
            let has_expected = hints.iter().any(|h| h == "expected");
            if !has_expected { return None; }
            if is_xss_noise_url(match location { Some(l) => l, None => "" }) { /* location не про
URL страницы, оставим как есть */ }
        }
        VectorKind::Sqli => {
            if !(has_errkw || has_5xx || has_delta) { return None; }
            if (400..=499).contains(&status) && !(has_delta || has_5xx) { return None; }
        }
        VectorKind::PathTraversal => {
            let has_expected = hints.iter().any(|h| h == "expected");
            if !(has_expected || has_5xx) { return None; }
            if status == 404 && !has_expected { return None; }
            if !has_expected && (has_delta || has_echo) && !has_5xx { return None; }
        }
        VectorKind::OpenRedirect => {
            let mut strong = false;
            if (300..=399).contains(&status) {
                if let Some(loc) = location {
                    let ll = loc.to_ascii_lowercase();
                    if (ll.starts_with("http://") || ll.starts_with("https://")) &&
ll.contains(&payload.to_ascii_lowercase()) {
                        strong = true;
                    }
                }
```

```
            } else {
                strong = hints.iter().any(|h| h == "meta-refresh" || h == "js-redirect");
            }
            if !strong { return None; }
        }
        VectorKind::CmdInjection => {
            let has_expected = hints.iter().any(|h| h == "expected");
            if !(has_expected || has_5xx) { return None; }
        }
        VectorKind::Ssti => {
            let has_expected = hints.iter().any(|h| h == "expected");
            if !(has_expected && (has_delta || has_5xx)) { return None; }
        }
    }

    let has_expected = hints.iter().any(|h| h == "expected");
    let format_evidence = |hs: &Vec<String>| -> String {
        if hs.is_empty() { "[]".to_string() } else {
            let joined = hs.iter().map(|h| format!("\"{}\"", h)).collect::<Vec<_>>().join(", ");
            format!("[{}]", joined)
        }
    };
    let (risk, ev) = if has_5xx && (has_echo || has_errkw || has_loc || has_delta || has_expected) {
        ("high".to_string(), format_evidence(&hints))
    } else if has_5xx || has_errkw || has_loc || has_delta || has_expected {
        ("medium".to_string(), format_evidence(&hints))
    } else {
        ("low".to_string(), format_evidence(&hints))
    };
    Some((risk, ev))
}

fn matches_skip_url(raw_url: &str, skip_paths: &[String]) -> bool {
    if skip_paths.is_empty() { return false; }
    if let Ok(url) = Url::parse(raw_url) {
        let path = url.path();
        let path_q = match url.query() { Some(q) => format!("{}?{}", path, q), None =>
path.to_string() };
        for s in skip_paths {
            let mut raw = s.trim();
            if let Some((head, _frag)) = raw.split_once('#') { raw = head.trim(); }
            if raw.is_empty() { continue; }
            let pat = if raw.starts_with('/') { raw.to_string() } else { format!("/{}", raw) };
            if pat.contains('?') {
                if path_q.starts_with(&pat) { return true; }
            } else {
                if path.starts_with(&pat) { return true; }
            }
        }
    }
    false
}

pub async fn analyze_with_threads(
    report: &ReconReport,
    headers: &[(String, String)],
    cookies: &[(String, String)],
    skip_paths: &Vec<String>,
    http_threads: usize,
) -> Result<Vec<DynamicIssue>, String> {
    let db_sets = load_fuzz_datasets_from_db().await;
    let mut hmap = HeaderMap::new();
    for (k, v) in headers {
        if let (Ok(name), Ok(val)) = (HeaderName::try_from(k.as_str()), HeaderValue::from_str(v)) {
            hmap.insert(name, val);
        }
    }
    if !cookies.is_empty() {
        let cookie_str = cookies
            .iter()
            .map(|(n, v)| format!("{}={}", n, v))
            .collect::<Vec<_>>()
            .join("; ");
        if let Ok(val) = HeaderValue::from_str(&cookie_str) {
            hmap.insert(COOKIE, val);
        }
    }

    let client = reqwest::Client::builder()
```

```
        .default_headers(hmap)
        .timeout(std::time::Duration::from_secs(10))
        .redirect(reqwest::redirect::Policy::limited(5))
        .build()
        .map_err(|e| e.to_string())?;

    let mut issues: Vec<DynamicIssue> = Vec::new();
    let urls: Vec<String> = report
        .sitemap
        .pages
        .iter()
        .filter(|p| (200..=399).contains(&p.status))
        .map(|p| p.url.clone())
        .collect();

    fn all_vectors() -> Vec<VectorKind> {
        vec![
            VectorKind::Xss,
            VectorKind::Sqli,
            VectorKind::OpenRedirect,
            VectorKind::PathTraversal,
            VectorKind::CmdInjection,
            VectorKind::Ssti,
        ]
    }

    use std::collections::HashSet as StdHashSet;
    let db_supported: StdHashSet<VectorKind> = match &db_sets {
        Some(map) => map.keys().cloned().collect(),
        None => all_vectors().into_iter().collect(),
    };


    let mut vectors_to_test: Vec<VectorKind> = all_vectors();
    {
        let mut set: StdHashSet<VectorKind> = StdHashSet::new();
        vectors_to_test.retain(|vk| set.insert(vk.clone()));
    }

    for url in urls {
        if matches_skip_url(&url, skip_paths) { continue; }
        let mut baseline_status: u16 = 0;
        let mut baseline_len: usize = 0;
        let mut baseline_text: String = String::new();
        if let Ok(resp) = client.get(&url).send().await {
            baseline_status = resp.status().as_u16();
            let text = resp.text().await.unwrap_or_default();
            baseline_len = text.len();
            baseline_text = text;
        }
        let base_param_names: Vec<String> = if let Ok(u) = Url::parse(&url) {
            u.query_pairs()
                .map(|(k, _)| k.to_string())
                .collect::<Vec<_>>()
        } else { Vec::new() };
        let candidate_params = if base_param_names.is_empty() { vec!["q".to_string()] } else {
base_param_names };

        #[derive(Clone)]
        struct Job {
            test_url: String,
            payload_for_classify: String,
            vector: VectorKind,
            param_name: Option<String>,
            expected: Vec<String>,
            risk_level: Option<String>,
        }

        let mut jobs: Vec<Job> = Vec::new();
        for vector in &vectors_to_test {
            let datasets = pick_datasets_for(vector, &db_sets);
            for ds in datasets {
                if !ds.methods.iter().any(|m| m.eq_ignore_ascii_case("GET")) { continue; }
                if ds.param_styles.iter().any(|s| s == "query") {
                    let mut pnames = candidate_params.clone();
                    let mut add: Vec<&str> = Vec::new();
                    match vector {
                        VectorKind::OpenRedirect => { add =
vec!["url","next","redirect","return","to","dest","continue","u"]; }
```

```
                        VectorKind::PathTraversal => { add =
vec!["file","path","page","dir","include","template"]; }
                        VectorKind::CmdInjection => { add =
vec!["cmd","exec","command","ping","ip","host"]; }
                        VectorKind::Ssti => { add = vec!["template","name","q","search","message"];
}
                        _ => {}
                    }
                    for k in add { if !pnames.iter().any(|s| s == k) { pnames.push(k.to_string()); }
}
                    for payload in ds.payloads.iter().take(10) {
                        for pname in &pnames {
                            for variant in build_value_variants(vector, payload) {
                                if let Some(test_url) = build_test_urls(&url, pname, &variant) {
                                    if matches_skip_url(&test_url, skip_paths) { continue; }
                                    jobs.push(Job {
                                        test_url,
                                        payload_for_classify: variant.clone(),
                                        vector: vector.clone(),
                                        param_name: Some(pname.clone()),
                                        expected: ds.expected.clone(),
                                        risk_level: ds.risk_level.clone(),
                                    });
                                }
                            }
                        }
                    }
                }
                if ds.param_styles.iter().any(|s| s == "path") {
                    for payload in ds.payloads.iter().take(5) {
                        if let Some(test_url) = inject_payload_in_path(&url, payload) {
                            if matches_skip_url(&test_url, skip_paths) { continue; }
                            jobs.push(Job {
                                test_url,
                                payload_for_classify: payload.clone(),
                                vector: vector.clone(),
                                param_name: None,
                                expected: ds.expected.clone(),
                                risk_level: ds.risk_level.clone(),
                            });
                        }
                    }
                }
            }
        }
        let client_ref = &client;
        let mut stream = stream::iter(jobs.into_iter().map(|job| {
            let client = client_ref.clone();
            let baseline_text_clone = baseline_text.clone();
            async move {
                match client.get(&job.test_url).send().await {
                    Ok(resp) => {
                        let status = resp.status().as_u16();
                        let location = resp.headers().get(reqwest::header::LOCATION).and_then(|v|
v.to_str().ok()).map(|s| s.to_string());
                        let body = resp.text().await.unwrap_or_default();
                        if let Some((risk0, evidence)) = classify(&job.vector, status, &body,
&job.payload_for_classify, location.as_deref(), baseline_status, baseline_len, &job.expected,
Some(&baseline_text_clone)) {
                            let risk = job.risk_level.clone().unwrap_or(risk0);
                            if !matches!(job.vector, VectorKind::Xss) ||
!is_xss_noise_url(&job.test_url) {
                                return Some(DynamicIssue { url: job.test_url, parameter:
job.param_name.clone(), vector: job.vector.clone(), risk, evidence, status, method:
"GET".to_string() });
                            }
                        }
                        None
                    }
                    Err(_) => None,
                }
            }
        }))
        .buffer_unordered(std::cmp::max(1, http_threads));

        while let Some(res) = stream.next().await {
            if let Some(issue) = res { issues.push(issue); }
        }
    }
```

```
    for page in &report.sitemap.pages {
        if page.forms.is_empty() { continue; }
        for form in &page.forms {
            if matches_skip_url(&form.action, skip_paths) { continue; }
            let vectors = vectors_to_test.clone();
            let fields = &form.fields;
            let fname_candidates: Vec<String> = if fields.is_empty() { vec!["q".to_string()] } else
{ fields.clone() };
            let action = &form.action;

            for vector in &vectors {
                let datasets = pick_datasets_for(vector, &db_sets);
                for ds in datasets {
                    if form.method.eq_ignore_ascii_case("POST") {
                        if !ds.methods.iter().any(|m| m.eq_ignore_ascii_case("POST")) { continue; }
                        if !ds.param_styles.iter().any(|s| s == "form") { continue; }
                        for payload in ds.payloads.iter().take(8) {
                            for fname in &fname_candidates {
                                let (b_status, b_len, b_text) = {
                                    let mut status: u16 = 0;
                                    let mut len: usize = 0;
                                    let mut text: String = String::new();
                                    let mut ser =
url::form_urlencoded::Serializer::new(String::new());
                                    for f in fields { ser.append_pair(f, "test"); }
                                    if let Ok(resp) = client.post(action)
                                        .header(reqwest::header::CONTENT_TYPE, "application/x-www-
form-urlencoded")
                                        .body(ser.finish())
                                        .send().await {
                                        status = resp.status().as_u16();
                                        let t = resp.text().await.unwrap_or_default();
                                        len = t.len();
                                        text = t;
                                    }
                                    (status, len, text)
                                };
                                for variant in build_value_variants(vector, payload) {
                                    let body = build_form_body(fields, fname, &variant);
                                    match client
                                        .post(action)
                                        .header(reqwest::header::CONTENT_TYPE, "application/x-www-
form-urlencoded")
                                        .body(body.clone())
                                        .send().await {
                                        Ok(resp) => {
                                            let status = resp.status().as_u16();
                                            let location =
resp.headers().get(reqwest::header::LOCATION).and_then(|v| v.to_str().ok()).map(|s| s.to_string());
                                            let text = resp.text().await.unwrap_or_default();
                                            if let Some((risk0, evidence)) = classify(vector,
status, &text, &variant, location.as_deref(), b_status, b_len, &ds.expected, Some(&b_text)) {
                                                let risk = ds.risk_level.clone().unwrap_or(risk0);
                                                if !matches!(vector, VectorKind::Xss) ||
!is_xss_noise_url(action) {
                                                    issues.push(DynamicIssue { url: action.clone(),
parameter: Some(fname.clone()), vector: vector.clone(), risk, evidence, status, method:
"POST".to_string() });
                                                }
                                            }
                                        }
                                        Err(_err) => {
                                        }
                                    }
                                }
                            }
                        }
                    } else {
                        if !ds.methods.iter().any(|m| m.eq_ignore_ascii_case("GET")) { continue; }
                        if !ds.param_styles.iter().any(|s| s == "form" || s == "query") { continue;
}
                        for payload in ds.payloads.iter().take(8) {
                            for fname in &fname_candidates {
                                let (b_status, b_len, b_text) = {
                                    let mut status: u16 = 0;
                                    let mut len: usize = 0;
                                    let mut text: String = String::new();
                                    let params: Vec<(String, String)> = fields.iter().map(|f|
```

```
(f.clone(), "test".to_string())).collect();
                                    if let Some(u) = build_get_with_params(action, &params) {
                                        if matches_skip_url(&u, skip_paths) { (status, len,
text.clone()) } else {
                                            if let Ok(resp) = client.get(&u).send().await {
                                                status = resp.status().as_u16();
                                                let t = resp.text().await.unwrap_or_default();
                                                len = t.len();
                                                text = t;
                                            }
                                            (status, len, text)
                                        }
                                    }
                                    else { (status, len, text) }
                                };
                                for variant in build_value_variants(vector, payload) {
                                    let params: Vec<(String, String)> = fields.iter().map(|f| {
                                        if f == fname { (f.clone(), variant.clone()) } else {
(f.clone(), "test".to_string()) }
                                    }).collect();
                                    if let Some(test_url) = build_get_with_params(action, &params) {
                                        if matches_skip_url(&test_url, skip_paths) { continue; }
                                        match client.get(&test_url).send().await {
                                            Ok(resp) => {
                                                let status = resp.status().as_u16();
                                                let location =
resp.headers().get(reqwest::header::LOCATION).and_then(|v| v.to_str().ok()).map(|s| s.to_string());
                                                let body = resp.text().await.unwrap_or_default();
                                                if let Some((risk0, evidence)) = classify(vector,
status, &body, &variant, location.as_deref(), b_status, b_len, &ds.expected, Some(&b_text)) {
                                                    let risk =
ds.risk_level.clone().unwrap_or(risk0);
                                                    if !matches!(vector, VectorKind::Xss) ||
!is_xss_noise_url(&test_url) {
                                                        issues.push(DynamicIssue { url: test_url,
parameter: Some(fname.clone()), vector: vector.clone(), risk, evidence, status, method:
"GET".to_string() });
                                                    }
                                                }
                                            }
                                            Err(_err) => { }
                                        }
                                    }
                                }
                            }
                        }
                    }
                }
            }
        }

    use std::collections::HashMap as StdHashMap;
    fn vec_to_str(v: &VectorKind) -> &'static str {
        match v { VectorKind::Xss => "XSS", VectorKind::Sqli => "SQLi", VectorKind::OpenRedirect =>
"OpenRedirect", VectorKind::PathTraversal => "PathTraversal", VectorKind::CmdInjection => "CMDi",
VectorKind::Ssti => "SSTI" }
    }
    fn norm_url_no_fragment(u: &str) -> String {
        if let Ok(mut p) = Url::parse(u) {
            p.set_fragment(None);
            p.to_string()
        } else { u.to_string() }
    }
    fn normalize_url_for_param(u: &str, param: Option<&str>) -> String {
        if let Ok(mut url) = Url::parse(u) {
            url.set_fragment(None);
            if let Some(pname) = param {
                let mut new_q: Vec<(String, String)> = Vec::new();
                for (k, v) in url.query_pairs() {
                    if k == pname {
                        new_q.push((k.to_string(), String::new()));
                    } else {
                        new_q.push((k.to_string(), v.to_string()));
                    }
                }
                url.query_pairs_mut().clear();
                for (k, v) in new_q { url.query_pairs_mut().append_pair(&k, &v); }
            }
```

```
            return url.to_string();
        }
        u.to_string()
    }
    fn risk_rank(r: &str) -> u8 { match r.to_ascii_lowercase().as_str() { "high" => 3, "medium" =>
2, "low" => 1, _ => 0 } }
    let mut best: StdHashMap<String, DynamicIssue> = StdHashMap::new();
    for it in issues.into_iter() {
        let norm_for_param = normalize_url_for_param(&it.url, it.parameter.as_deref());
        let key = format!(
            "{}|{}|{}|{}",
            it.method.to_ascii_uppercase(),
            norm_for_param,
            it.parameter.clone().unwrap_or_default(),
            vec_to_str(&it.vector)
        );
        if let Some(cur) = best.get_mut(&key) {
            if risk_rank(&it.risk) > risk_rank(&cur.risk) {
                *cur = it;
            }
        } else {
            best.insert(key, it);
        }
    }
    let mut out: Vec<DynamicIssue> = best.into_values().collect();
    out.sort_by(|a, b| a.url.cmp(&b.url).then_with(|| a.method.cmp(&b.method)).then_with(||
a.parameter.cmp(&b.parameter)));
    Ok(out)
}

pub async fn analyze(
    report: &ReconReport,
    headers: &[(String, String)],
    cookies: &[(String, String)],
    skip_paths: &Vec<String>,
) -> Result<Vec<DynamicIssue>, String> {
    let def = std::thread::available_parallelism().map(|n| n.get()).unwrap_or(4);
    analyze_with_threads(report, headers, cookies, skip_paths, std::cmp::max(1, def)).await
}

fn inject_payload_in_path(base: &str, payload: &str) -> Option<String> {
    if let Ok(mut u) = Url::parse(base) {
        let mut segments: Vec<String> = u
            .path_segments()
            .map(|it| it.map(|s| s.to_string()).collect::<Vec<_>>())
            .unwrap_or_default();
        let has_non_empty = segments.iter().any(|s| !s.is_empty());
        if !has_non_empty {
            return None;
        } else {
            if let Some(last) = segments.last_mut() {
                if last.is_empty() { segments.push(payload.to_string()); } else { *last =
payload.to_string(); }
            }
        }
        let joined: Vec<&str> = segments.iter().filter_map(|s| if s.is_empty() { None } else {
Some(s.as_str()) }).collect();
        if joined.is_empty() { return None; }
        let new_path = format!("/{}", joined.join("/"));
        u.set_path(&new_path);
        Some(u.to_string())
    } else { None }
}
```

download_cve_tests.rs

```
use std::io::Write;
use std::path::Path;

use tempfile::tempdir;

use scanner::download_cve;

#[tokio::test]
async fn fetch_and_extract_cve() {
    use std::time::Instant;

    let tmp = tempdir().expect("tempdir");
```

```
    let dest = tmp.path().join("cvelistV5");

    let t0 = Instant::now();
    let extracted = download_cve::fetch_and_extract(download_cve::DEFAULT_CVELIST_URL, &dest)
        .await
        .expect("fetch_and_extract ok");
    let first_elapsed = t0.elapsed();

    assert!(extracted.exists(), "extracted dir must exist");
    let marker = dest.join(".extracted");
    assert!(marker.exists(), "marker .extracted must be created");


    let inner = download_cve::find_inner_root(&dest).expect("inner directory should exist after
unzip");
    assert!(inner.is_dir(), "inner root must be a directory");
    let mut has_entries = false;
    if let Ok(mut rd) = std::fs::read_dir(&inner) {
        has_entries = rd.next().is_some();
    }
    assert!(has_entries, "inner directory should contain files or subdirectories");
    println!("CVE download+extract time (first run): {:?}", first_elapsed);
}


#[tokio::test]
async fn fetch_and_extract_nvd_all_years() {
    use chrono::Datelike;

    let now = chrono::Utc::now();
    let y_to = now.year();
    let y_from = 2002;

    let tmp = tempdir().expect("tempdir");

    for y in y_from..=y_to {
        let dest = tmp.path().join(format!("nvd-{}", y));
        let url = download_cve::NVD20_URL_TMPL.replace("{year}", &format!("{}", y));

        let extracted = download_cve::fetch_and_extract(&url, &dest)
            .await
            .expect(&format!("fetch_and_extract (NVD {}) ok", y));

        assert!(extracted.exists(), "extracted dir must exist for {}", y);
        let marker = dest.join(".extracted");
        assert!(marker.exists(), "marker .extracted must be created for NVD {}", y);

        let json_name = format!("nvdcve-2.0-{}.json", y);
        let json_path = dest.join(&json_name);
        assert!(json_path.is_file(), "NVD JSON must exist after extraction: {}", json_name);
    }
}

#[allow(dead_code)]
fn build_test_zip_with_inner_dir() -> Vec<u8> {
    let mut buf: Vec<u8> = Vec::new();
    {
        use zip::write::SimpleFileOptions;
        let cursor = std::io::Cursor::new(&mut buf);
        let mut zip = zip::ZipWriter::new(cursor);
        let options = SimpleFileOptions::default();

        zip.start_file("cvelistV5-main/README.txt", options).unwrap();
        zip.write_all(b"hello").unwrap();

        zip.finish().unwrap();
    }
    buf
}

#[test]
fn detect_inner_root_in_zip() {
    use std::io::Cursor;
    let tmp = tempdir().expect("tempdir");
    let dest = tmp.path().join("cvelistV5");
    std::fs::create_dir_all(&dest).expect("create dest");

    {
        let cursor = Cursor::new(bytes);
```

```
        let mut zip = zip::ZipArchive::new(cursor).expect("zip open");
        zip.extract(&dest).expect("zip extract");
    }

    let inner = dest.join("cvelistV5-main");
    assert!(inner.is_dir(), "inner directory must exist: {}", inner.display());
    let readme = inner.join("README.txt");
    assert!(readme.is_file(), "README.txt must exist after extraction");

    let found = download_cve::find_inner_root(&dest);
    assert!(found.is_some(), "find_inner_root should find inner directory");
    assert_eq!(found.unwrap(), inner);
}

#[test]
fn returns_none_if_no_inner_dirs() {
    let tmp = tempdir().unwrap();
    let root = tmp.path();

    std::fs::write(root.join("file.txt"), b"x").unwrap();

    let res = download_cve::find_inner_root(root);
    assert!(res.is_none(), "no inner directories should return None");
}

#[test]
fn default_out_dir_is_relative() {
    let p = download_cve::default_out_dir();
    assert!(p.ends_with(Path::new("downloads/cvelistV5")));
    assert!(p.is_relative(), "default_out_dir should be relative path");
}
```

import_cve_tests.rs

```
use scanner::import_cve::test_api as api;

#[test]
fn pick_description_prefers_english() {
    let descs = vec![(Some("fr"), "bonjour"), (Some("en"), "hello")];
    assert_eq!(api::pick_description_from(descs).as_deref(), Some("hello"));

    let descs = vec![(Some("de"), "hallo"), (None, "generic")];
    assert_eq!(api::pick_description_from(descs).as_deref(), Some("hallo"));

    let descs: Vec<(Option<&str>, &str)> = vec![];
    assert!(api::pick_description_from(descs).is_none());
}

#[test]
fn public_exploit_detected_by_tag_or_url() {
    assert!(api::has_public_exploit_from(
        Some("https://example.com/advisory"),
        &["Exploit"]
    ));

    assert!(api::has_public_exploit_from(
        Some("https://host/path/poc.txt"),
        &[]
    ));

    assert!(!api::has_public_exploit_from(Some("https://host/info"), &[]));
}

#[test]
fn collect_recommendations_prefers_marked() {
    let cna = vec![
        (Some("https://vendor.com/security/advisory-1"), vec![]),
        (Some("https://vendor.com/blog"), vec![]),
    ];
    let adp = vec![
        (Some("https://thirdparty.com/patch"), vec![]),
        (Some("https://misc.com/news"), vec![]),
    ];
    let urls = api::collect_recommendations_from(cna, adp);
    assert!(urls.iter().any(|u| u.contains("advisory")));
    assert!(urls.iter().any(|u| u.contains("patch")));
    assert!(!urls.iter().any(|u| u.contains("blog")));
```

```
    let cna2 = vec![(Some("https://vendor.com/page1"), vec![])];
    let adp2 = vec![(Some("https://third.com/page"), vec![])];
    let urls2 = api::collect_recommendations_from(cna2, adp2);
    assert_eq!(urls2, vec!["https://vendor.com/page1".to_string()]);
}

#[test]
fn first_cwe_returns_first_entry() {
    let got = api::first_cwe_from(vec![Some("CWE-79")]);
    assert_eq!(got.as_deref(), Some("CWE-79"));

    let got2 = api::first_cwe_from(vec![None]);
    assert!(got2.is_none());
}

#[test]
fn pick_cvss_prefers_v3_1_then_v3_0() {
    let (s1, sev1) = api::pick_cvss_from(
        Some((Some(9.1), Some("CRITICAL"))),
        None,
    );
    assert_eq!(s1, Some(9.1));
    assert_eq!(sev1.as_deref(), Some("CRITICAL"));

    let (s2, sev2) = api::pick_cvss_from(
        None,
        Some((Some(7.5), Some("HIGH"))),
    );
    assert_eq!(s2, Some(7.5));
    assert_eq!(sev2.as_deref(), Some("HIGH"));

    let (s3, sev3) = api::pick_cvss_from(None, None);
    assert!(s3.is_none());
    assert!(sev3.is_none());
}

#[test]
fn extract_cve_id_returns_id_or_none() {
    let p = "/root/cves/2024/12xxx/CVE-2024-12345.json";
    assert_eq!(api::extract_cve_id(p).as_deref(), Some("CVE-2024-12345"));

    let p2 = "/root/invalid.json";
    assert!(api::extract_cve_id(p2).is_none());
}
```

base_crawler_tests.rs

```
use httpmock::prelude::*;
use scanner::crawler::test_api as api;

#[tokio::test]
async fn validate_target_extracts_headers_and_meta() {
    let server = MockServer::start();

    let index_html = r#"
        <html><head>
          <meta name=generator content="WordPress 6.0">
        </head><body>Hello</body></html>
    "#;

    server.mock(|when, then| {
        when.method(GET).path("/");
        then.status(200)
            .header("Server", "nginx")
            .header("X-Powered-By", "PHP/8.2")
            .body(index_html);
    });

    let info = api::validate(&server.base_url()).await.expect("validate ok");

    assert_eq!(info.status, 200);
    assert_eq!(info.server.as_deref(), Some("nginx"));
    assert_eq!(info.x_powered_by.as_deref(), Some("PHP/8.2"));
    assert_eq!(info.generator.as_deref(), Some("WordPress 6.0"));
}

#[tokio::test]
async fn validate_handles_missing_headers_and_missing_meta() {
```

```
    let server = MockServer::start();

    server.mock(|when, then| {
        when.method(GET).path("/");
        then.status(200).body("<html><head></head><body>OK</body></html>");
    });

    let info = api::validate(&server.base_url()).await.expect("validate ok");

    assert_eq!(info.status, 200);
    assert!(info.server.is_none());
    assert!(info.x_powered_by.is_none());
    assert!(info.generator.is_none());
}

#[tokio::test]
async fn validate_extracts_generator_from_variants() {
    let server = MockServer::start();

    let html = r#"
        <meta content="Drupal 10.x" name="generator">
        <meta NAME=GENERATOR CONTENT='Joomla 5.0'>
    "#;

    server.mock(|when, then| {
        when.method(GET).path("/");
        then.status(200).body(html);
    });

    let info = api::validate(&server.base_url()).await.expect("validate ok");

    assert!(
        info.generator.is_some(),
        "Expected generator from variant meta tags"
    );
}


#[tokio::test]
async fn passive_recon_crawls_same_origin_and_derives_vectors() {
    let server = MockServer::start();

    let index_html = format!(
        r#"<a href="/a?search=test&id=5">A</a>
           <a href="{}">Off</a>
           <a href="/redir?url=https://evil.example">R</a>"#,
        "https://external.example/"
    );

    server.mock(|when, then| {
        when.method(GET).path("/");
        then.status(200).header("Server", "Apache").body(index_html);
    });

    server.mock(|when, then| {
        when.method(GET).path("/a");
        then.status(200).body("<a href=\"/b?cmd=ls\">B</a>");
    });

    server.mock(|when, then| {
        when.method(GET).path("/b");
        then.status(200).body("<a href=\"/c?file=../../etc/passwd\">C</a>");
    });

    server.mock(|when, then| {
        when.method(GET).path("/c");
        then.status(200).body("done");
    });

    let report = api::recon(&server.base_url(), 10).await.expect("recon ok");

    assert!(report
        .discovered_urls
        .iter()
        .all(|u| u.starts_with(&server.base_url())));

    assert!(report.discovered_urls.iter().any(|u| u.contains("/a")));
    assert!(report.discovered_urls.iter().any(|u| u.contains("/b")));
```

```
    use scanner::crawler::VectorKind as V;
    let kinds: std::collections::HashSet<_> = report.vectors.iter().cloned().collect();

    assert!(kinds.contains(&V::Xss));
    assert!(kinds.contains(&V::Sqli));
    assert!(kinds.contains(&V::OpenRedirect));
    assert!(kinds.contains(&V::CmdInjection));
    assert!(kinds.contains(&V::PathTraversal));
}


#[tokio::test]
async fn passive_recon_respects_max_depth() {
    let server = MockServer::start();

    server.mock(|when, then| {
        when.method(GET).path("/");
        then.status(200)
            .body("<a href=\"/a\">A</a>");
    });

    server.mock(|when, then| {
        when.method(GET).path("/a");
        then.status(200)
            .body("<a href=\"/b\">B</a>");
    });

    server.mock(|when, then| {
        when.method(GET).path("/b");
        then.status(200)
            .body("<a href=\"/c\">C</a>");
    });

    let report = api::recon(&server.base_url(), 1).await.expect("recon");


    assert!(report.discovered_urls.iter().any(|u| u.ends_with("/")));
    assert!(!report.discovered_urls.iter().any(|u| u.contains("/a")));
}


#[tokio::test]
async fn passive_recon_ignores_non_http_links() {
    let server = MockServer::start();

    server.mock(|when, then| {
        when.method(GET).path("/");
        then.status(200).body(
            r#"
            <a href="mailto:test@example.com">mail</a>
            <a href="javascript:alert(1)">js</a>
            <a href="tel:+123456">tel</a>
            <a href="/ok">OK</a>
        "#,
        );
    });

    server.mock(|when, then| {
        when.method(GET).path("/ok");
        then.status(200).body("OK");
    });

    let report = api::recon(&server.base_url(), 5).await.expect("recon");

    assert!(report.discovered_urls.iter().any(|u| u.contains("/ok")));
    assert!(!report.discovered_urls.iter().any(|u| u.contains("mailto")));
    assert!(!report.discovered_urls.iter().any(|u| u.contains("javascript")));
    assert!(!report.discovered_urls.iter().any(|u| u.contains("tel")));
}
```

crawler_headers_tests.rs

```
use httpmock::prelude::*;

use scanner::crawler::test_api as api;

#[tokio::test]
async fn crawler_sends_custom_headers_and_cookies() {
```

```
    let server = MockServer::start();

    let _root = server.mock(|when, then| {
        when.method(GET)
            .path("/")
            .header("X-Debug", "1")
            .header_exists("Cookie");
        then.status(200)
            .header("Server", "nginx")
            .body("<a href=\"/a?x=1\">A</a>");
    });

    let _a = server.mock(|when, then| {
        when.method(GET)
            .path("/a")
            .header("X-Debug", "1")
            .header_exists("Cookie");
        then.status(200).body("done");
    });

    let headers = vec![("X-Debug".to_string(), "1".to_string())];
    let cookies = vec![
        ("session".to_string(), "abc".to_string()),
        ("feature".to_string(), "on".to_string()),
    ];

    let report = api::recon_with(&server.base_url(), 5, headers, cookies)
        .await
        .expect("recon_with ok");

    assert!(report.discovered_urls.iter().any(|u| u.ends_with('/')));
    assert!(report.discovered_urls.iter().any(|u| u.contains("/a")));
}
```

crawler_signatures_tests.rs

```
use httpmock::prelude::*;
use scanner::crawler::test_api as api;
#[tokio::test]
async fn detects_technologies_versions_and_cpes() {
    let server = MockServer::start();

    let body = r#"
        <html>
          <head>
            <meta name="generator" content="WordPress 6.5.2">
            <script src="/3.6.0/jquery.min.js"></script>
          </head>
          <body>Hello</body>
        </html>
    "#;

    let _root = server.mock(|when, then| {
        when.method(GET).path("/");
        then.status(200)
            .header("Server", "nginx/1.25.0")
            .header("X-Jenkins", "2.452.1")
            .body(body);
    });

    let report = api::recon(&server.base_url(), 3).await.expect("recon ok");
    let techs = report.technologies.join(" | ");

    assert!(techs.contains("Jenkins"));
    assert!(techs.contains("WordPress 6.5.2"));
    assert!(techs.to_lowercase().contains("nginx"));
    assert!(techs.contains("jQuery 3.6.0"));

    let cpes = report.cpes;
    assert!(cpes.iter().any(|c| c.contains("jenkins:jenkins")));
    assert!(cpes.iter().any(|c| c.contains("nginx:nginx")));
}

#[tokio::test]
async fn detects_nothing_on_empty_response() {
    let server = MockServer::start();

    server.mock(|when, then| {
```

```
        when.method(GET).path("/");
        then.status(200).body("");
    });

    let report = api::recon(&server.base_url(), 1).await.expect("recon ok");

    assert!(report.technologies.is_empty(), "expected no techs");
    assert!(report.cpes.is_empty(), "expected no cpes");
}

#[tokio::test]
async fn detects_technologies_from_headers_only() {
    let server = MockServer::start();

    server.mock(|when, then| {
        when.method(GET).path("/");
        then.status(200)
            .header("Server", "Apache/2.4.57")
            .header("X-Powered-By", "PHP/8.2.1")
            .body("OK");
    });

    let report = api::recon(&server.base_url(), 1).await.expect("recon ok");
    let techs = report.technologies.join(" | ");

    assert!(techs.contains("Apache 2.4.57"));
    assert!(techs.contains("PHP 8.2.1"));
}

#[tokio::test]
async fn detects_technologies_from_html_only() {
    let server = MockServer::start();

    let body = r#"
        <html>
            <head>
                <meta name="generator" content="Drupal 10.1">
                <script src="/4.0.0/vue.js"></script>
            </head>
        </html>
    "#;

    server.mock(|when, then| {
        when.method(GET).path("/");
        then.status(200).body(body);
    });

    let report = api::recon(&server.base_url(), 1).await.expect("ok");
    let techs = report.technologies.join(" | ");

    assert!(techs.contains("Drupal 10.1"));
    assert!(techs.contains("Vue.js 4.0.0"));
}

#[tokio::test]
async fn crawler_follows_links_with_depth() {
    let server = MockServer::start();

    server.mock(|when, then| {
        when.method(GET).path("/");
        then.status(200)
            .body(r#"<a href="/next">next</a>"#);
    });

    server.mock(|when, then| {
        when.method(GET).path("/next");
        then.status(200).header("Server", "lighttpd/1.4.75");
    });

    let report = api::recon(&server.base_url(), 2).await.expect("ok");

    let techs = report.technologies.join(" | ");
    assert!(techs.contains("lighttpd 1.4.75"), "crawler did not follow next page");
}
```

ІЛЮСТРАТИВНА ЧАСТИНА

МЕТОД ТА ЗАСІБ АНАЛІЗУ БЕЗПЕКИ ВЕБСАЙТІВ

Виконав: студент 2 курсу групи 1БС-24 м спеціальності 125 Кібербезпека та захист інформації

_____ Іван КРАВЧУК

Керівник: ~~к. т. н.,~~ доц., доцент каф. ЗІ

_____ Юрій БАРИШЕВ

«19» грудня 2025 р.

АНАЛІЗ ЗАСОБІВ ТЕСТУВАННЯ НА ПРОНИКНЕННЯ

Засіб	Опис
Acunetix	Автоматизований сканер вебзастосунків, який швидко виявляє вразливості, забезпечуючи детальні звіти з рекомендаціями щодо виправлення проблем. Підтримує автоматизацію сканування для великих проєктів і має низький рівень помилкових результатів [18].
Aircrack-ng	Інструмент для аналізу безпеки Wi-Fi мереж, що дозволяє тестувати протоколи WEP/WPA/WPA2 шляхом злому ключів, перехоплення пакетів і моніторингу трафіку [19]. Він ефективний для перевірки бездротових мереж, але потребує спеціального обладнання.
Astra Pentest	Платформа, що поєднує автоматизоване та ручне тестування вебзастосунків, надаючи детальні звіти для відповідності стандартам, таким як PCI DSS. Підходить для компаній, які потребують комплексного аналізу безпеки з підтримкою командної роботи [20].
Burp Suite	Потужний інструмент для перехоплення HTTP-трафіку, аналізу вебзастосунків і гнучкого тестування з підтримкою плагінів для розширення функціоналу. Він підтримує як автоматизоване, так і ручне тестування, що робить його популярним серед професіоналів безпеки [21].
DirBuster	Інструмент для пошуку прихованих директорій і файлів на вебсерверах через brute-force із підтримкою словників і графічного інтерфейсу [22]. Він ефективний для розвідки, але застарілий порівняно з сучасними альтернативами.
Ffuf	Швидкий fuzzer для тестування вебзастосунків, що дозволяє гнучко налаштовувати словники для пошуку вразливостей у параметрах, шляхах чи API [23].
Gobuster	Інструмент для швидкого пошуку директорій і файлів на серверах через brute-force, із підтримкою словників і відкритим кодом [24]. Він оптимізує швидкість розвідки, але обмежений пошуком шляхів.
Hashcat	Високопродуктивний інструмент для злому хешів паролів, підтримує широкий спектр алгоритмів. Він ефективний для офлайн-атак, але потребує потужного обладнання [25].
Hydra	Інструмент для brute-force атак на паролі через протоколи, такі як SSH, FTP, HTTP, що забезпечує високу швидкість тестування [26]. Підходить для перевірки слабких паролів, але може викликати блокування IP.
Intruder	Хмарний інструмент для автоматизованого тестування вебзастосунків і fuzzing, з акцентом на простоту використання та швидкість [27]. Підходить для швидкого аналізу, але має обмежені налаштування.
John the Ripper	Інструмент із відкритим кодом для злому паролів на CPU, підтримує різні формати хешів і словникові атаки. Ефективний для простих сценаріїв, але повільніший за GPU-альтернативи [28].
Kismet	Інструмент для моніторингу Wi-Fi мереж, що дозволяє виявляти пристрої, аналізувати трафік і працювати з дронами. Корисний для розвідки бездротових мереж, але потребує спеціального обладнання [29].
Metasploit	Фреймворк для створення, тестування та виконання експлойтів із великою базою модулів для різних типів атак [30]. Підтримує командну роботу і є стандартом для тестування на проникнення.
Nikto	Сканер для швидкого аналізу конфігурацій вебсерверів, що виявляє базові вразливості, такі як застарілі версії чи неправильні налаштування [31]. Простий у використанні, але обмежений для динамічних додатків.

Засіб	Опис
Nmap	Інструмент для розвідки мереж і сканування портів із підтримкою скриптів для аналізу служб і вразливостей. Базовим інструментом для мережевої розвідки [32].
Nuclei	Сканер на основі шаблонів для швидкого виявлення вразливостей, підтримує YAML-конфігурації та відкритий код. Він дозволяє налаштувати тести, але потребує ручного налаштування шаблонів [33].
OWASP ZAP	Безкоштовний інструмент із відкритим кодом для активного та пасивного сканування вебзастосунків, підтримуваний спільнотою [34]. Підходить для початківців, але може генерувати помилкові результати.
RustScan	Швидкий сканер портів із відкритим кодом, що інтегрується з Nmap для ефективної розвідки мереж. Потребує додаткових інструментів для аналізу [35].
sqlmap	Інструмент для автоматизованого тестування SQL-ін'єкцій, що підтримує різні системи керування базами даних. Ефективний для спеціалізованих атак, але обмежений SQL-сценаріями [36].
Wireshark	Аналізатор мережевого трафіку, що забезпечує детальний моніторинг і фільтрацію пакетів для виявлення аномалій. Він корисний для пасивного аналізу, але потребує ручної перевірки даних [37].

АНАЛІЗ СКАНЕРІВ ВРАЗЛИВОСТЕЙ

Засіб	Опис
Greenbone	Безкоштовна платформа з вебінтерфейсом для сканування вразливостей у мережах, що підтримує налаштування і аналіз середніх за розміром проєктів. Інтегрується з базами вразливостей і підходить для організацій із обмеженим бюджетом, забезпечуючи регулярний аудит безпеки [48].
Nessus	Потужний сканер вразливостей із великою базою даних, що підтримує масштабованість, інтеграцію з SIEM і детальні звіти для великих мережевих інфраструктур [49]. Ефективний для планового аудиту і відповідності стандартам безпеки, таким як PCI DSS.
Nexpose	Інструмент для сканування мереж із визначення пріоритетності ризиків, інтеграцією з InsightVM і підтримкою великих мережевих інфраструктур. Дозволяє ефективно керувати вразливостями в складних середовищах, але потребує значних ресурсів [50].
OpenSCAP	Безкоштовний інструмент для перевірки конфігурацій операційних систем, що підтримує стандарти NIST/DISA. Ефективний для аналізу системного рівня, але обмежений для мережевих перевірок і потребує ручного налаштування [51].
OpenVAS	Безкоштовний сканер із відкритим кодом, що підтримує налаштування та регулярне оновлення бази вразливостей [52]. Підходить для базового аудиту, але менш точний порівняно з комерційними рішеннями.
Qualys VMDR	Хмарна платформа для управління вразливостями, що підтримує AWS/Azure, визначення пріоритетності ризиків і детальні звіти для великих мереж. Оптимізує автоматизований аудит, але залежить від хмарної інфраструктури [53].
QualysGuard	Хмарний інструмент для автоматизованого сканування великих мереж, з підтримкою відповідності стандартам і генерацією детальних звітів. Ефективний для великих організацій, але має обмежену гнучкість для специфічних сценаріїв [54].
Rapid7 InsightVM	Інструмент для управління вразливостями, що пропонує визначення пріоритетності ризиків, інтеграцію з SIEM і asset management для великих мереж [55]. Підходить для складних інфраструктур, але потребує ретельного налаштування.
Retina Network Scanner	Сканер для середніх мереж із підтримкою відповідності стандартам і автоматизованими звітами, орієнтований на виявлення системних вразливостей [56].
Tenable.io	Хмарна платформа для сканування вразливостей із підтримкою AWS/Azure, автоматизацією і відповідністю стандартам. Оптимальна для хмарних середовищ, але залежить від постачальника [57].
Tripwire IP360	Інструмент для enterprise-систем, що забезпечує управління активами, пріоритизацію ризиків і аудит великих мереж. Підходить для складних інфраструктур, але складний для малого бізнесу [58].
Wiz	Хмарний сканер для швидкого аналізу безпеки в AWS/Azure/GCP, з акцентом на відповідність стандартам і захист хмарних середовищ [59].

МАТЕМАТИЧНИЙ ОПИС

Множина всіх елементів системи, що підлягають перевірці, позначається як:

$$X = \{x_1, x_2, \dots, x_n, \}$$

де x_i – окремий компонент вебсайту, який аналізується під час процесу сканування. Кожен елемент цієї множини може мати власні характеристики наприклад, бібліотеки які використовуються, конфігураційні файли чи API запити.

Результатом аналізу є множина виявлених вразливостей V, яка є підмножиною множини всіх відомих вразливостей V_{all}:

$$V \subseteq V_{all}$$

Множина V_{all} формується на основі бази відомих уразливостей, яка містить записи про типові помилки конфігурації, логічні помилки, вразливості OWASP Top 10.

Кожна знайдена вразливість $v_i \in V$ характеризується певним рівнем критичності m_i, що визначається відповідно до впорядкованої множини:

$$M = \{Low, Medium, Hight, Critical\}$$

Ця шкала базується на загальноприйнятому стандарті CVSS [68], який використовується у світовій практиці для оцінки небезпеки вразливостей. Критерії розподілу базуються на таких параметрах, як вплив на конфіденційність, цілісність і доступність даних, а також складність експлуатації.

Для знаходження вразливостей використовується множина баз даних, які містять інформацію для аналізу:

$$S = \{D_V, D_F\}$$

де D_V – база відомих вразливостей, що містить опис, сигнатури, умови виникнення та способи виявлення;

D_F – база даних для фазингу, що зберігає набори тестових даних, параметри запитів і сценарії для виявлення неочевидних або логічних помилок у поведінці системи.

Для підтримки актуальності бази вразливостей виконується її синхронізація з відкритими джерелами вразливостей:

$$update: S = S'$$

Основний процес методу полягає у перетворенні множини вхідних компонентів X у множину знайдених вразливостей V з урахуванням даних, що зберігаються у внутрішніх базах:

$$analyze: X \times S \to V$$

У ході цього процесу здійснюється сканування компонентів, виконання тестових запитів, перевірка відповідей сервера, а також пошук вразливих бібліотек.

Процес фазингу відповідає за автоматичну відправку тестових даних і використовується для дослідження реакції системи на нестандартні вхідні параметри:

$$fuzz: D_F \times X \to V$$

Математична модель для аналізу безпеки вебсайтів має наступний вигляд:

$$\{X, V, S, M, analyze(X, S), fuzz(D_F, X), update(S)\}$$

МЕТОД АНАЛІЗУ БЕЗПЕКИ ВЕБСАЙТІВ

Крок 1. Ініціалізація та синхронізація баз даних.

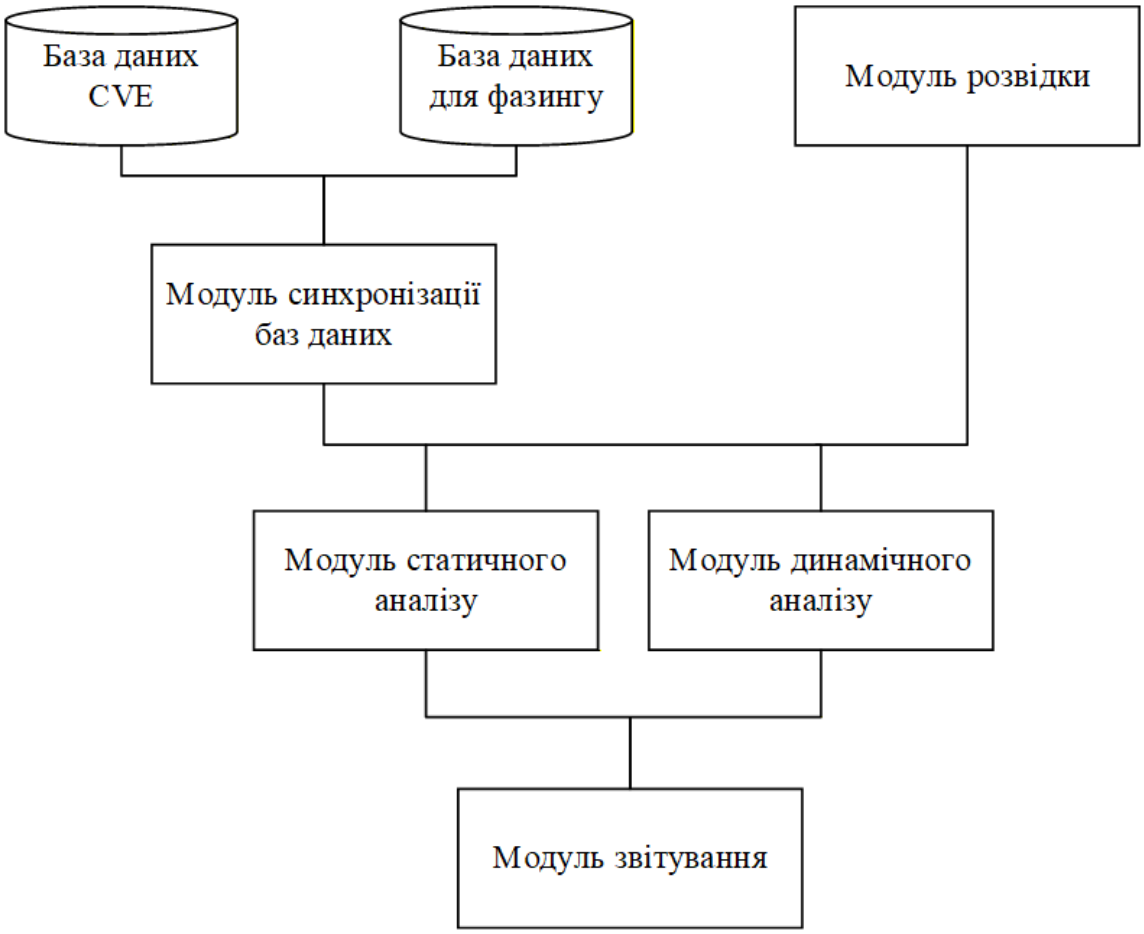
Крок 2. Пасивна розвідка структури вебсайту та виявлення компонентів.

Крок 3. Статичний аналіз виявлених компонентів.

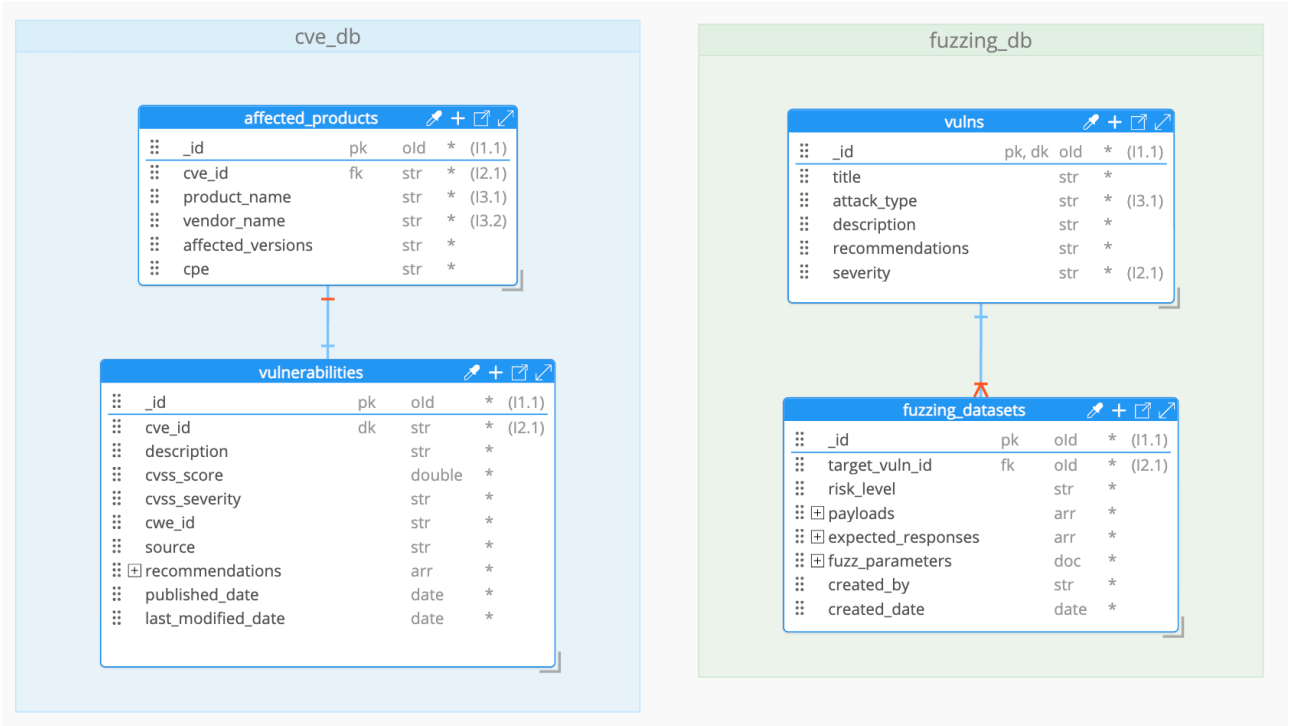
Крок 4. Динамічний аналіз шляхом активного тестування та фазингу.

Крок 5. Формування звіту.

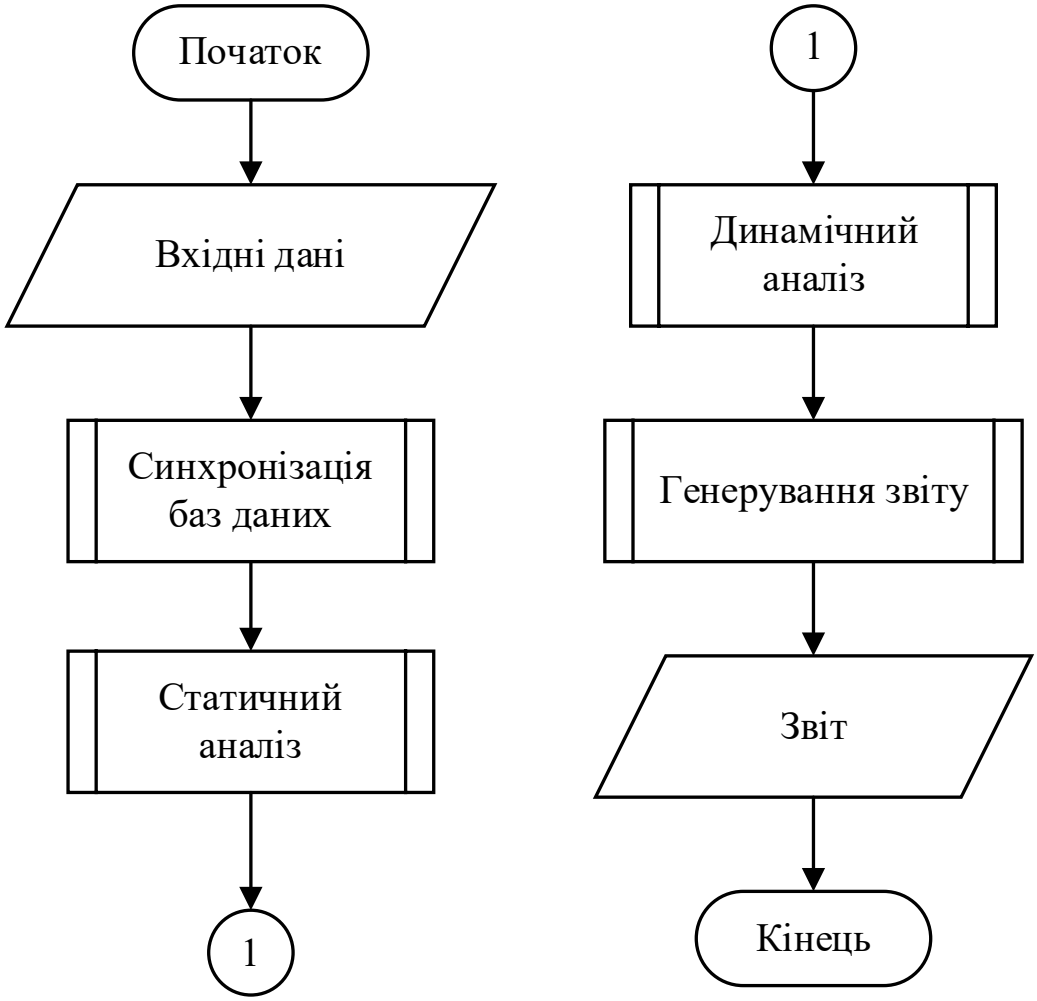
АРХІТЕКТУРА ЗАСОБУ АНАЛІЗУ БЕЗПЕКИ ВЕБСАЙТІВ



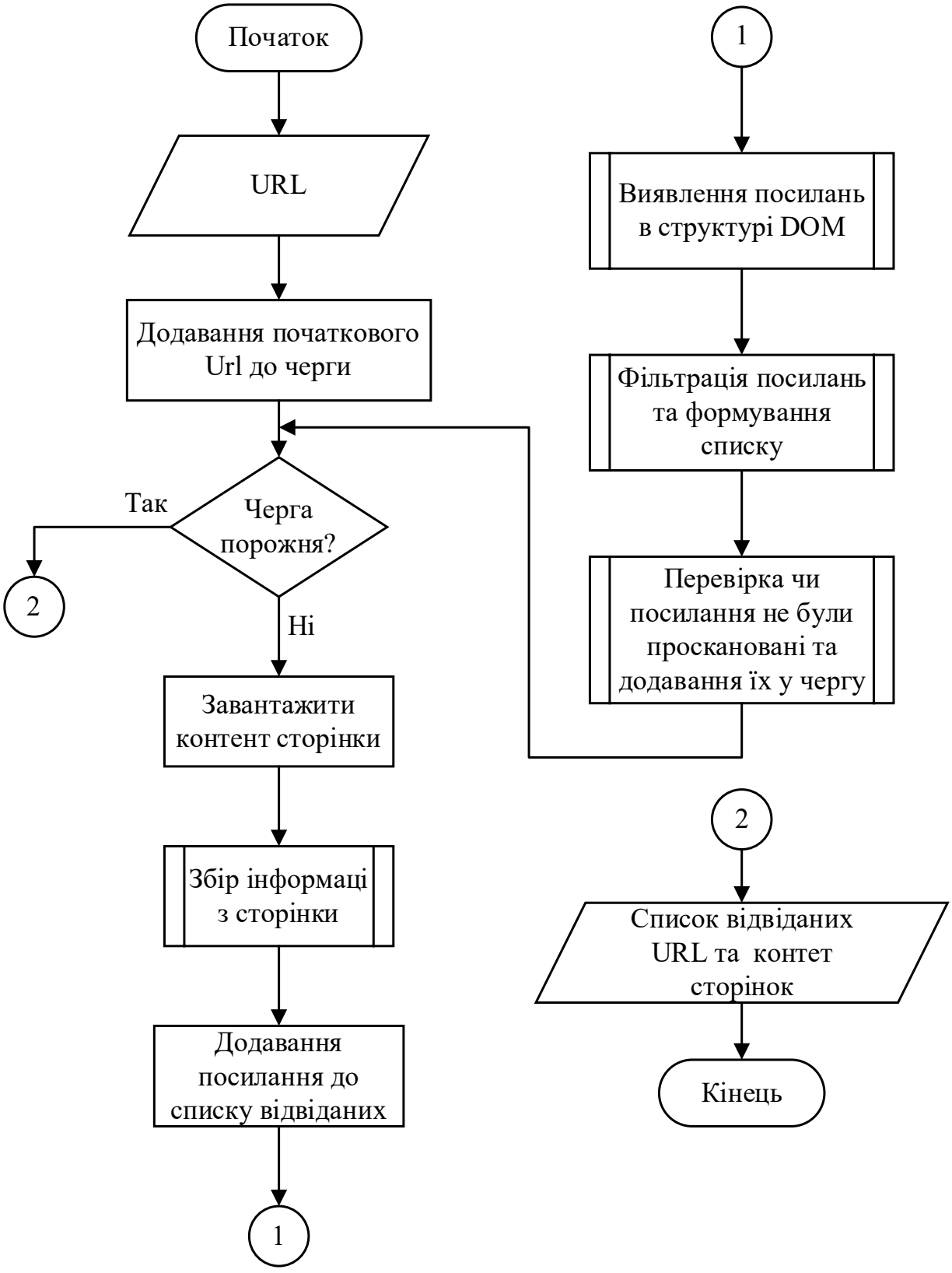
МОДЕЛЬ БАЗИ ДАНИХ ВРАЗЛИВОСТЕЙ



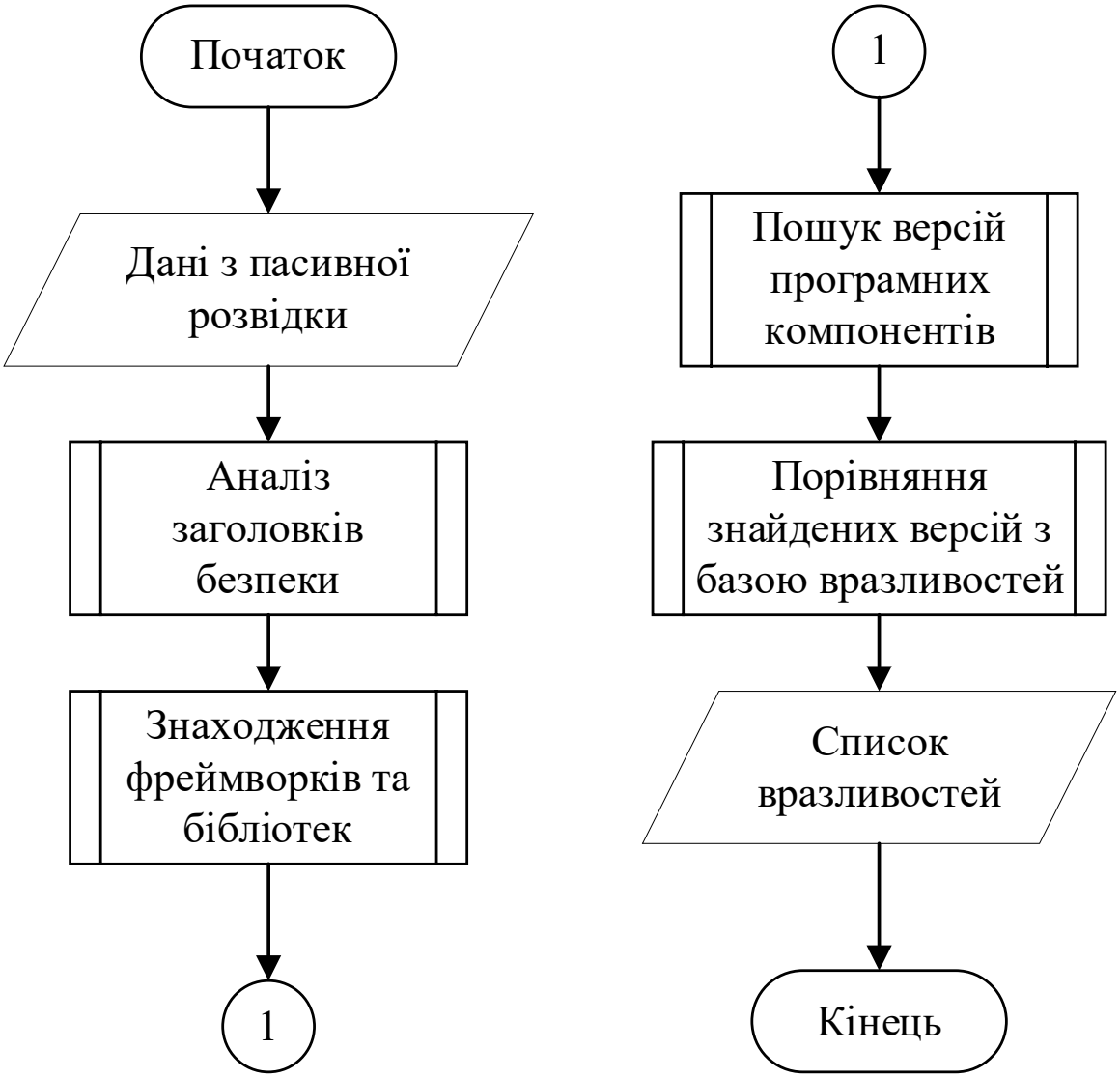
УЗАГАЛЬНЕНИЙ АЛГОРИТМ РОБОТИ ЗАСОБУ



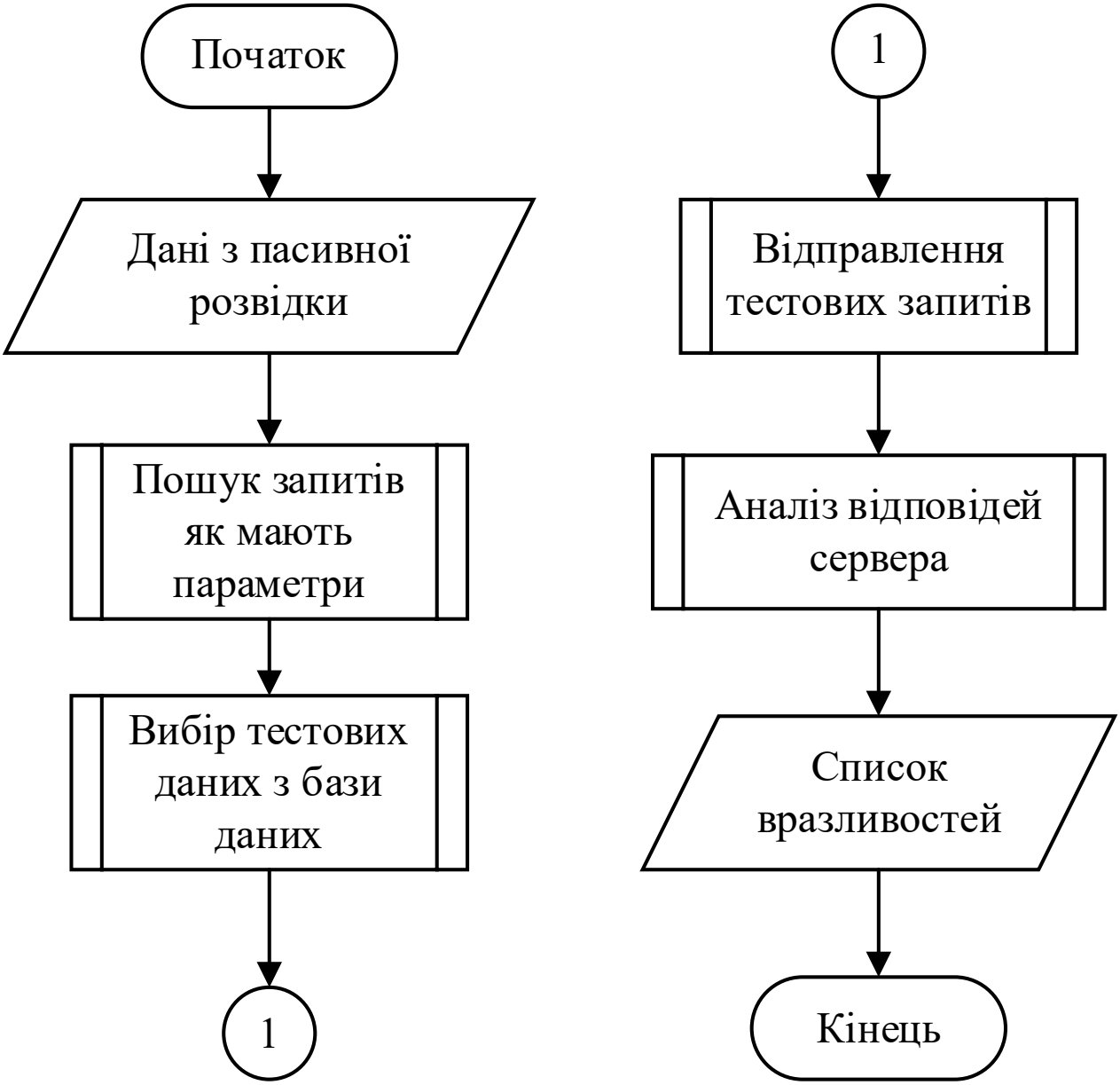
АЛГОРИТМ ПАСИВНОЇ РОЗВІДКИ



АЛГОРИТМ СТАТИЧНОГО АНАЛІЗУ



АЛГОРИТМ ДИНАМІЧНОГО АНАЛІЗУ



ФОРМАТИ ЗВІТІВ

```
ЗВІТ ПРО СКАНУВАННЯ БЕЗПЕКИ
Ціль: https://example.com
Дата сканування: {date}
Версія сканера: 2.0

SQL Injection (CWE-89)
CVSS: HIGH

Scope:
• https://example.com/search?query=test

Description:
Вразливість дозволяє виконувати довільні SQL-запити через неналежну валідацію вхідних даних.
Може призвести до витоку конфіденційної інформації з бази даних.

Recommendation:
1. Використовувати параметризовані запити
2. Застосувати валідацію вхідних даних
3. Обмежити привілеї облікового запису БД
```

Макет звіту у форматі TXT

```xml
<security_report>
    <scan_info>
        <target_url>https://example.com</target_url>
        <scan_date>{data}</scan_date>
        <scanner_version>2.0</scanner_version>
    </scan_info>
    <vulnerabilities>
        <vulnerability>
            <name>SQL Injection (CWE-89)</name>
            <cvss_score>8.6</cvss_score>
            <cvss_severity>HIGH</cvss_severity>
            <scope>
                <item>https://example.com/search?query=test</item>
            </scope>
            <description>Вразливість дозволяє виконувати довільні SQL-запити через неналежну валідацію вхідних даних.
            Може призвести до витоку конфіденційної інформації з бази даних.</description>
            <recommendation>
                1. Використовувати параметризовані запити
                2. Застосувати валідацію вхідних даних
                3. Обмежити привілеї облікового запису БД
            </recommendation>
        </vulnerability>
    </vulnerabilities>
</security_report>
```

Макет звіту у форматі XML

SQL Injection (CWE-89)

CVSS: HIGH

Scope:

- https://example.com/search?query=test

Description:

Вразливість дозволяє виконувати довільні SQL-запити через неналежну валідацію вхідних даних. Може призвести до витоку конфіденційної інформації з бази даних.

Recommendation:

1. Використовувати параметризовані запити

2. Застосувати валідацію вхідних даних

3. Обмежити привілеї облікового запису БД

Макет звіту у форматі PDF

РЕЗУЛЬТАТИ МОДУЛЬНОГО ТЕСТУВАННЯ

```
running 11 tests
test cve_tests::default_out_dir_is_relative ... ok
test cve_tests::extract_cve_id_returns_id_or_none ... ok
test cve_tests::pick_cvss_prefers_v3_1_then_v3_0 ... ok
test cve_tests::first_cwe_returns_first_entry ... ok
test cve_tests::pick_description_prefers_english ... ok
test cve_tests::public_exploit_detected_by_tag_or_url ... ok
test cve_tests::collect_recommendations_prefers_marked ... ok
test cve_tests::returns_none_if_no_inner_dirs ... ok
test cve_tests::detect_inner_root_in_zip ... ok
test cve_tests::fetch_and_extract_nvd_all_years ... ok
test cve_tests::fetch_and_extract_cve has been running for over 60 seconds
test cve_tests::fetch_and_extract_cve ... ok

test result: ok. 11 passed; 0 failed; 0 ignored; 0 measured; 0 filtered out; finished in 155.13s
```

Результати модульного тестування роботи з базами даних CVE

```
running 3 tests
test fuzzing_tests::defaults_sets_are_non_empty ... ok
test fuzzing_tests::merge_deduplicates_and_sorts ... ok
test fuzzing_tests::download_all_fuzzing_datasets ... ok

test result: ok. 3 passed; 0 failed; 0 ignored; 0 measured; 0 filtered out; finished in 7.16s
```

Результати модульного тестування завантаження тестових наборів

```
running 12 tests
test cravler_tests::base::validate_target_extracts_headers_and_meta ... ok
test cravler_tests::base::validate_extracts_generator_from_variants ... ok
test cravler_tests::base::validate_handles_missing_headers_and_missing_meta ... ok
test cravler_tests::signatures::detects_nothing_on_empty_response ... ok
test cravler_tests::base::passive_recon_respects_max_depth ... ok
test cravler_tests::base::passive_recon_ignores_non_http_links ... ok
test cravler_tests::headers::crawler_sends_custom_headers_and_cookies ... ok
test cravler_tests::signatures::detects_technologies_from_html_only ... ok
test cravler_tests::signatures::detects_technologies_from_headers_only ... ok
test cravler_tests::signatures::crawler_follows_links_with_depth ... ok
test cravler_tests::base::passive_recon_crawls_same_origin_and_derives_vectors ... ok
test cravler_tests::signatures::detects_technologies_versions_and_cpes ... ok

test result: ok. 12 passed; 0 failed; 0 ignored; 0 measured; 0 filtered out; finished in 0.14s
```

Результати тестування модуля пасивної розвідки

```
running 5 tests
test static_tests::basic::static_analyze_returns_empty_when_no_cpes ... ok
test static_tests::cpe_edge_cases::static_analyze_early_return_without_cpes ... ok
test static_tests::basic::static_analyze_survives_no_db_and_returns_empty ... ok
test static_tests::cpe_edge_cases::static_analyze_accepts_edge_case_versions_tokens ... ok
test static_tests::cpe_edge_cases::static_analyze_survives_large_cpe_list has been running for over 60 seconds
test static_tests::cpe_edge_cases::static_analyze_survives_large_cpe_list ... ok

test result: ok. 5 passed; 0 failed; 0 ignored; 0 measured; 0 filtered out; finished in 100.98s
```

Результати тестування модуля статичного аналізу

```
running 4 tests
test dynamic_tests::dynamic_analyze_returns_empty_on_safe_pages ... ok
test dynamic_tests::dynamic_analyze_respects_skip_paths ... ok
test dynamic_tests::dynamic_analyze_respects_headers_and_cookies ... ok
test dynamic_tests::dynamic_analyze_scans_all_statuses_including_4xx_5xx ... ok

test result: ok. 4 passed; 0 failed; 0 ignored; 0 measured; 0 filtered out; finished in 1.51s
```

Результати тестування модуля динамічного аналізу

РЕЗУЛЬТАТИ РОБОТИ ЗАСОБУ

```
macbook@MacBook-Pro debug % ./scanner
Usage:
  scanner --update-cve                        Download and import CVE (cvelistV5)
  scanner --update-nvd [--years Y[,..]|all]   Download and import NVD 2.0 for years (default: current, previous)
  scanner --update-cves                       Force re-download and re-import (both sources)
  scanner --update-fuzz                       Seed fuzzing DB from SecLists and defaults
  scanner --url <url> [--header H:V] [--cookie n=v|"a=b; c=d"] [--skip-path p1[,p2]] [--max-pages N] [--max-depth N] [--stay-domain true|false] [--dynamic] [-v|--verbose]
         [--report-format txt[,xml[,pdf]]] [--report-file <path_without_ext>] [--log-file <path>]

Notes:
  - You can repeat --header and --cookie multiple times, or pass multiple cookies in one string separated by ';'.
  - To exclude paths from crawling, use --skip-path (repeatable or comma-separated).
  - Control crawl depth and size with --max-depth and --max-pages.
  - By default the crawler stays within the domain. To change: --stay-domain false.
  - --update-nvd downloads the current and previous year by default; specify a list with --years or use --years all.
  - -v/--verbose enables detailed output.
  - Reporting: provide --report-format (txt,xml,pdf) and --report-file as base path; example: --report-format txt,xml --report-file ./out/report
```

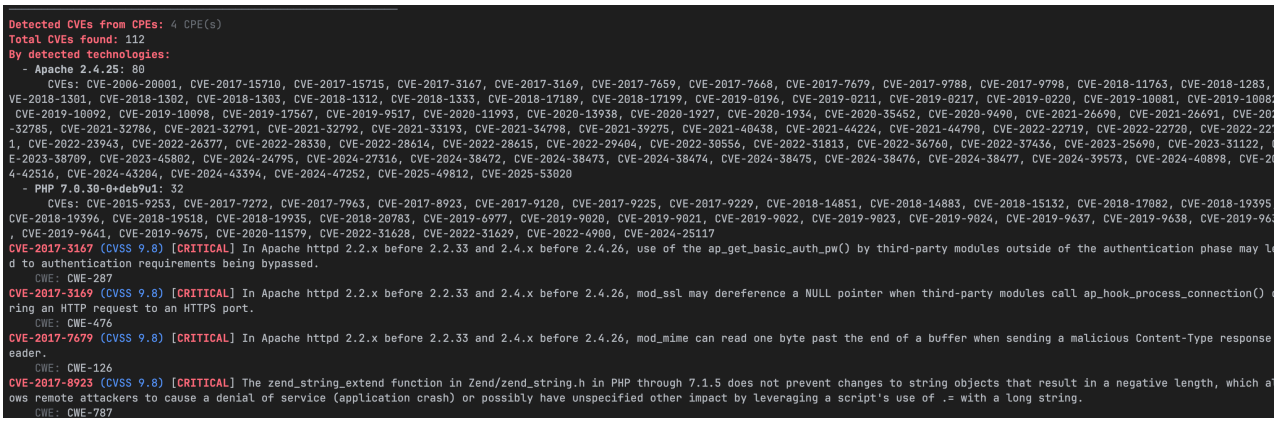
Інтерфейс командного рядка

```
macbook@MacBook-Pro debug % ./scanner --url 'http://localhost/' --skip-path '/logout.php,/setup.php' --cookie 'PHPSESSID=i431t2pejh2l2lpvvjdcu5th67' --report-format txt,xml,pdf --report-file test --dynamic
▶ Starting passive reconnaissance for http://localhost/
Target: http://localhost/ → http://localhost/ (status 200)
Server: Apache/2.4.25 (Debian)
Discovered pages: 31
Technologies: Apache 2.4.25, PHP 7.0.30-0+deb9u1, Server: Apache/2.4.25 (Debian)
Threads: 16
──────────────────────────────────────────────────

GET URLs:
200      7KB http://localhost/
200     15KB http://localhost/instructions.php
200      4KB http://localhost/vulnerabilities/brute/
200      4KB http://localhost/vulnerabilities/exec/
200      4KB http://localhost/vulnerabilities/csrf/
200      4KB http://localhost/vulnerabilities/fi/?page=include.php
200      4KB http://localhost/vulnerabilities/upload/
200      5KB http://localhost/vulnerabilities/captcha/
200      4KB http://localhost/vulnerabilities/sqli/
200      5KB http://localhost/vulnerabilities/sqli_blind/
```

Запуск сканування DVWA з параметрами

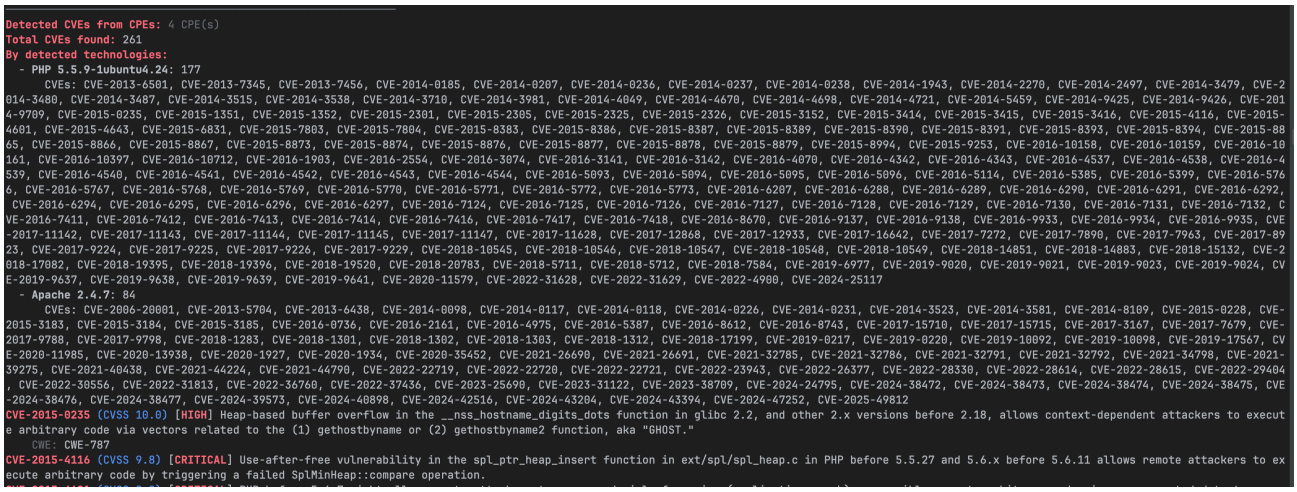
```
──────────────────────────────────────────────────

POST URLs:
POST http://localhost/security.php (params: seclev_submit, security, user_token
POST http://localhost/security.php?test=%22%3E%3Cscript%3Eeval(window.name)%3C/script%3E (params: seclev_submit, security, user_token
POST http://localhost/vulnerabilities/brute/ (params: Login, password, user_token, username
POST http://localhost/vulnerabilities/captcha/ (params: Change, password_conf, password_current, password_new, step, user_token
POST http://localhost/vulnerabilities/csp/
POST http://localhost/vulnerabilities/exec/ (params: Submit, ip, user_token
POST http://localhost/vulnerabilities/javascript/ (params: phrase, send, token
POST http://localhost/vulnerabilities/upload/ (params: MAX_FILE_SIZE, Upload, uploaded, user_token
POST http://localhost/vulnerabilities/weak_id/
POST http://localhost/vulnerabilities/xss_s/ (params: btnClear, btnSign, mtxMessage, txtName, user_token
──────────────────────────────────────────────────

Parameterized URLs:
200      4KB http://localhost/vulnerabilities/fi/?page=include.php
200     15KB http://localhost/instructions.php?doc=readme
200      3KB http://localhost/instructions.php?doc=PDF
200     11KB http://localhost/instructions.php?doc=changelog
200     39KB http://localhost/instructions.php?doc=copying
200     12KB http://localhost/instructions.php?doc=PHPIDS-license
200      4KB http://localhost/vulnerabilities/fi/?page=file1.php
200      4KB http://localhost/vulnerabilities/fi/?page=file2.php
200      4KB http://localhost/vulnerabilities/fi/?page=file3.php
200      5KB http://localhost/security.php?test=%22%3E%3Cscript%3Eeval(window.name)%3C/script%3E
──────────────────────────────────────────────────

Resources by type:

JS:
http://localhost/dvwa/js/add_event_listeners.js
http://localhost/vulnerabilities/csp/source/impossible.js
http://localhost/dvwa/js/dvwaPage.js

CSS:
http://localhost/dvwa/css/main.css

HTML:
200      7KB http://localhost/
200     15KB http://localhost/instructions.php
200      4KB http://localhost/vulnerabilities/brute/
200      4KB http://localhost/vulnerabilities/exec/
200      4KB http://localhost/vulnerabilities/csrf/
```

Результати пасивної розвідки DVWA



Результати динамічного аналізу вразливостей DVWA

Результати динамічного тестування DVWA



Результати статичного сканування Hackazon

Результати динамічного сканування Hackazon

Security Report AUTO-GENERATED

Target: http://localhost/
Scan date: 2025-12-11
Scanner version: 1.0

CVE-2017-3167

CVE: CVE-2017-3167
CWE: CWE-287

CVSS: CRITICAL

Description:

In Apache httpd 2.2.x before 2.2.33 and 2.4.x before 2.4.26, use of the ap_get_basic_auth_pw() by third-party modules outside of the authentication phase may lead to authentication requirements being bypassed.

Recommendation:

- https://access.redhat.com/errata/RHSA-2017:3477
- http://www.debian.org/security/2017/dsa-3896
- https://security.gentoo.org/glsa/201710-32
- https://access.redhat.com/errata/RHSA-2017:3476
- https://access.redhat.com/errata/RHSA-2017:2479
- https://access.redhat.com/errata/RHSA-2017:2483
- https://support.apple.com/HT208221
- https://security.netapp.com/advisory/ntap-20180601-0002/
- https://access.redhat.com/errata/RHSA-2017:3193
- https://access.redhat.com/errata/RHSA-2017:2478
- https://access.redhat.com/errata/RHSA-2017:3195
- http://www.oracle.com/technetwork/security-advisory/cpuoct2017-3236626.html
- https://access.redhat.com/errata/RHSA-2017:3194
- https://access.redhat.com/errata/RHSA-2017:3475

Фрагмент PDF-звіту з результатами сканування DVWA

ПОКАЗНИКИ ЕКОНОМІЧНОЇ ЕФЕКТИВНОСТІ

Критерії	Експерт 1	Експерт 2	Експерт 3
1. Технічна здійсненність концепції	4	4	4
2. Ринкові переваги (наявність аналогів)	3	3	3
3. Ринкові переваги (ціна продукту)	4	4	4
4. Ринкові переваги (технічні властивості)	3	3	4
5. Ринкові переваги (експлуатаційні витрати)	4	4	4
6. Ринкові перспективи (розмір ринку)	4	4	4
7. Ринкові перспективи (конкуренція)	2	3	3
8. Практична здійсненність (наявність фахівців)	4	4	4
9. Практична здійсненність (наявність фінансів)	4	4	4
10. Практична здійсненність (необхідність нових матеріалів)	4	4	4
11. Практична здійсненність (термін реалізації)	4	4	4
12. Практична здійсненність (розробка документів)	4	4	4
Сума балів (СБі)	44	45	46
Середньоарифметична сума балів (СБс)	45		

$$ЗВ = \frac{453\ 339,46}{0,7} = 629\ 763,86$$

$$T_{ок} = 1 / 0,454 = 2,24 \text{ р.}$$