Міністерство освіти і науки України
Київський національний університет імені Тараса Шевченка

Факультет інформаційних технологій
Кафедра кібербезпеки та захисту інформації

ДОПУСТИТИ ДО ЗАХИСТУ:
В.о. завідувача кафедри
кібербезпеки та захисту
інформації
_____Іван ПАРХОМЕНКО
«___» червня 2025 р.

ПОЯСНЮВАЛЬНА ЗАПИСКА

кваліфікаційної роботи

галузь знань 12 Інформаційні технології
(шифр і назва галузі знань)
спеціальність 125 Кібербезпека
(код і назва спеціальності)
освітній ступень бакалавр
освітня програма Кібербезпека
(назва освітньо-професійної програми)
на тему: «Засіб захисту веб-додатків з використанням Burp Suite»

Виконавець: студент IV курсу, групи КБ-42

_____ **Дмитро ГАЙДАМАЧЕНКО**
(підпис) (ім'я, прізвище)

	Підпис	**Ім'я, прізвище**
Керівник		Яніна ШЕСТАК
Нормоконтроль		Іван БІЛОКОНЬ

Київ 2025

Міністерство освіти і науки України
Київський національний університет імені Тараса Шевченка

Факультет інформаційних технологій
Кафедра кібербезпеки та захисту інформації

ЗАТВЕРДЖЕНО:
В.о. завідувача кафедри
кібербезпеки
та захисту інформації
_____ Іван ПАРХОМЕНКО
«29» листопада 2024 р.

ЗАВДАННЯ
на виконання кваліфікаційної роботи

спеціальності 125 Кібербезпека
(код і назва спеціальності)

освітньої програми Кібербезпека
(назва освітньо-професійної програми)

студенту **КБ-42** **Гайдамаченку Дмитру Сергійовичу**
(група) (прізвище ім'я по-батькові)

Тема кваліфікаційної роботи Засіб захисту веб-додатків з використанням Burp Suite

1. ПІДСТАВИ ДЛЯ ПРОВЕДЕННЯ РОБОТИ

Тема кваліфікаційної роботи затверджена на засіданні кафедри кібербезпеки та захисту інформації протокол №6 від 28.11.2024 р.

2. ВИХІДНІ ДАНІ ДЛЯ ПРОВЕДЕННЯ РОБІТ

Середовище DVWA, інструмент Burp Suite, список OWASP Top 10, приклади атак на веб-додатки, типова архітектура клієнт-серверної взаємодії.

3. ЗМІСТ РОЗРАХУНКОВО-ПОЯСНЮВАЛЬНОЇ ЗАПИСКИ

Необхідно ознайомитися з основними вразливостями веб-додатків, провести аналіз OWASP Top 10, дослідити інструменти автоматизованого та ручного тестування безпеки, виконати практичне пентестування середовища DVWA за допомогою Burp Suite, виявити вразливості, запропонувати заходи їх усунення та оцінити ефективність впроваджених рішень.

4. ВИМОГИ ДО РЕЗУЛЬТАТІВ ВИКОНАННЯ РОБОТИ

Практична цінність виявлення і усунення вразливостей веб-додатка з використанням Burp Suite, побудова ефективної моделі перевірки безпеки та підвищення загального рівня захищеності веб-ресурсів.

5. ДАТА ВИДАЧІ ЗАВДАННЯ

Дата видачі завдання: 29 листопада 2024 року

Завдання видала _____ (підпис) Яніна ШЕСТАК (ініціали, прізвище)

Завдання прийняв до виконання _____ (підпис) Дмитро ГАЙДАМАЧЕНКО (ініціали, прізвище)

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Найменування етапів робіт	Строки виконання робіт (початок-кінець)	Відмітка про виконання
1	Уточнення постановки задачі	29.11.2024 – 22.01.2025	*виконано*
2	Аналіз літератури	29.01.2025 – 14.02.2025	*виконано*
3	Обґрунтування вибору рішення	15.02.2025 - 28.02.2025	*виконано*
4	Ознайомлення з інструментами тестування безпеки	29.02.2025 - 12.03.2025	*виконано*
5	Проведення практичного тестування веб-додатка	13.03.2025 – 28.03.2025	*виконано*
6	Виявлення вразливостей і впровадження заходів безпеки	01.04.2025 – 18.04.2025	*виконано*
7	Повторне тестування і оцінка результатів	20.04.2025 – 28.04.2025	*виконано*
8	Оформлення пояснювальної записки	08.05.2025 – 23.05.2025	*виконано*
9	Підготовка до захисту дипломної роботи	28.05.2025 – 13.06.2025	*виконано*

Завдання видала _____ (підпис) Яніна ШЕСТАК (ім'я, прізвище)

Завдання прийняв до виконання _____ (підпис) Дмитро ГАЙДАМАЧЕНКО (ім'я, прізвище)

Термін подання кваліфікаційної роботи до ЕК 13 червня 2025 року

РЕФЕРАТ

Пояснювальна записка кваліфікаційної роботи складається зі вступу, трьох розділів, загальних висновків, списку використаних джерел, додатків, має 68 сторінок основного тексту, 52 рисунки. Список використаних джерел містить 24 найменувань і займає 3 сторінки**.**

Метою роботи є комплексне дослідження вразливостей веб-додатків, оцінка ефективності методів їх виявлення та усунення, а також практична перевірка роботи інструменту Burp Suite у процесі тестування безпеки.

Для досягнення поставленої мети у роботі сформульовано наступні завдання:

- провести аналіз типових кіберзагроз, пов’язаних із веб-додатками;
- розглянути поширені вразливості OWASP Top 10;
- дослідити методи атак на веб-додатки;
- проаналізувати реальні інциденти безпеки, пов’язані з уразливостями;
- порівняти сучасні інструменти тестування безпеки;
- здійснити тестування веб-додатка DVWA за допомогою Burp Suite;
- розробити та впровадити заходи безпеки;
- оцінити ефективність впроваджених рішень після повторного тестування.

Об'єктом дослідження є інформаційна безпека веб-додатків в умовах сучасного цифрового середовища.

Предметом дослідження є методи та інструменти виявлення, аналізу і усунення вразливостей веб-додатків, зокрема за допомогою Burp Suite.

Практичною цінністю отриманих результатів є можливість застосування запропонованих методів тестування та заходів безпеки в реальних проєктах і навчанні фахівців з кібербезпеки.

Ключові слова: SQL-ін’єкція, XSS, Burp Suite, HTTP-запити, HTTPS, DVWA, OWASP Top 10, пентестинг, сесійні атаки, автоматизоване сканування.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ ТА СКОРОЧЕНЬ

CSP	–	Content Security Policy – політика безпеки вмісту
CSRF	–	Cross-Site Request Forgery – міжсайтове підроблення запитів
DVWA	–	Damn Vulnerable Web Application – навмисно вразливий веб-додаток
GET	–	HTTP-метод для запиту ресурсу
HSTS	–	HTTP Strict Transport Security – механізм примусового використання HTTPS
HTTP	–	HyperText Transfer Protocol – протокол передавання гіпертексту
HTTPS	–	HyperText Transfer Protocol Secure – захищений протокол HTTP
IDOR	–	Insecure Direct Object Reference – небезпечне пряме посилання на об’єкт
IPS	–	Intrusion Prevention System
MITM	–	Man-in-the-Middle – атака «людина посередині»
ORM	–	Object-Relational Mapping – об'єктно-реляційне відображення
OCTAVE	–	The Operationally Critical Threat, Asset, and Vulnerability Evaluation
(O)TD	–	(OWASP) Threat Dragon
OWASP	–	Open Web Application Security Project
PASTA	–	Process for Attack Simulation & Threat Analysis
POST	–	HTTP-метод для надсилання даних на сервер
RCE	–	Remote Code Execution – віддалене виконання коду
SQL	–	Structured Query Language – мова структурованих запитів
SQLi	–	SQL Injection – SQL-ін’єкція
SSTI	–	Server-Side Template Injection – ін’єкція шаблону на стороні сервера
SSRF	–	Server-Side Request Forgery – підробка серверного запиту
SSL	–	Secure Sockets Layer
TLS	–	Transport Layer Security
VPN	–	Virtual Private Network
XML	–	Extensible Markup Language
XSS	–	Cross-Site Scripting – міжсайтове скриптування

ВСТУП

У наш час веб-додатки стали важливою частиною майже всіх сфер — від бізнесу й державного управління до освіти, медицини та повсякденного життя. Через те, що до них можна зручно дістатись через інтернет, вони приваблюють не тільки звичайних користувачів, а й зловмисників. Із розвитком веб-технологій зростає й кількість загроз, пов’язаних із крадіжкою даних, порушенням конфіденційності, а також цілісності та доступності веб-ресурсів.

Незважаючи на те, що сучасні фреймворки та засоби розробки надають вбудовані механізми захисту, значна частина веб-додатків залишається вразливою до типових атак — таких як SQL-ін’єкції, міжсайтове скриптування (XSS), підробка запитів між сайтами (CSRF), атаки на ідентифікацію сесій тощо. Більшість цих вразливостей описано в OWASP Top 10 — авторитетному списку найкритичніших ризиків безпеки веб-додатків.

У зв’язку з цим особливої актуальності набувають інструменти для тестування безпеки, що дозволяють виявити та усунути вразливості ще на етапі розробки або під час експлуатації веб-додатка. Одним із найпотужніших і найпопулярніших рішень у цій сфері є Burp Suite — комплекс програмних модулів для перехоплення, модифікації, аналізу HTTP(S)-трафіку та автоматизованого сканування.

Актуальність теми пояснюється тим, що цілеспрямованих атак стає все більше, а захист веб-додатків часто впроваджують поверхнево або й взагалі нехтують ним. Усе це може призвести до витоку важливих даних. У роботі я хочу зосередитись саме на практичному боці питання — як реально перевірити безпеку веб-додатка за допомогою Burp Suite і на що звертати увагу при цьому.

Метою кваліфікаційної роботи є підвищення рівня безпеки веб-додатків шляхом виявлення та усунення вразливостей із використанням Burp Suite як основного інструменту тестування.

Практичне значення одержаних результатів полягає в тому, що запропоновані методи тестування безпеки можуть бути застосовані в реальних проєктах для перевірки надійності веб-додатків. Запроваджені заходи з усунення вразливостей (SQLi, XSS, незашифрована передача даних) можуть бути масштабовані на інші додатки з подібною архітектурою. Результати можуть бути корисними також для навчання майбутніх фахівців із кібербезпеки.

Тези стосовно важливості виявлення та усунення вразливостей у веб-додатках і підвищення ефективності їхнього тестування були апробовані на VIII Міжнародній науково-практичній конференції “Проблеми кібербезпеки інформаційно-комунікаційних систем” (PCSICS).

РОЗДІЛ 1
ОБ'ЄКТ ЗАХИСТУ: ЗАГРОЗИ ТА ВРАЗЛИВОСТІ ВЕБ-ДОДАТКІВ

1.1 Аналіз загроз веб-додатків у сучасних кіберзагрозах

У сучасну цифрову епоху веб-додатки стали звичним і навіть базовим елементом багатьох інформаційних систем. Вони дозволяють користувачам отримувати доступ до різноманітних сервісів, ресурсів і функцій у реальному часі, й саме тому активно використовуються в електронній комерції, державному секторі, фінансовій сфері, освіті та інших важливих галузях. Але через свою доступність і важливість веб-додатки все частіше стають цілями атак. Зловмисники намагаються скористатися їхніми слабкими місцями, щоб отримати несанкціонований доступ, змінити дані, впровадити шкідливий код або просто вивести систему з ладу [1, 2].

За даними звітів провідних аналітичних компаній, таких як Verizon (Data Breach Investigations Report), понад 40% зареєстрованих кіберінцидентів пов'язані саме з веб-додатками [3]. Вразливості у веб-сервісах часто стають початковою точкою компрометації інформаційної системи в цілому. Це зумовлено кількома факторами: відкритістю веб-інтерфейсів, складністю сучасних веб-архітектур, використанням сторонніх компонентів і бібліотек, а також людським фактором (недостатньо захищене програмне забезпечення, помилки конфігурації, недотримання політик безпеки) [4, 5].

Джерела загроз для веб-додатків можна поділити на такі категорії:

- Зовнішні атакуючі, які ініціюють атаки через публічні мережі, використовуючи інструменти сканування, автоматизовані експлойти, ботнети або цілеспрямовані ручні атаки [6].
- Внутрішні порушники, включаючи співробітників компанії, адміністраторів або партнерів, які зловживають доступом до ресурсів.

- Програмні вразливості, що виникають внаслідок помилок у коді, відсутності валідації даних, поганої обробки виключень, тощо.
- Сторонні компоненти, які є інтегрованими бібліотеками, фреймворками, API або плагінами, що можуть містити відомі уразливості або потенційно бути скомпрометованими [4, 5].

Ключові напрями реалізації загроз:

- Компрометація конфіденційності: перехоплення або викрадення особистої, фінансової або комерційної інформації (logins, паролі, банківські дані).
- Порушення цілісності: модифікація вмісту баз даних, підміна веб-сторінок, зміна бізнес-логіки застосунку.
- Порушення доступності: атаки типу відмови в обслуговуванні (DoS, DDoS), які призводять до недоступності веб-додатка або його окремих функцій.
- Захоплення сесій та підміна ідентифікаторів: що дозволяє здійснювати дії від імені легітимних користувачів без їхньої згоди [5, 7].

Актуальні типи загроз і приклади реалізації:

- SQL-ін’єкції (SQLi): впровадження шкідливих SQL-запитів для витоку або зміни даних;
- Cross-Site Scripting (XSS): виконання шкідливого скрипту в браузері жертви;
- Cross-Site Request Forgery (CSRF): примус користувача виконати небажані дії в контексті автентифікованої сесії;
- Broken Access Control: обхід обмежень доступу до об’єктів через URL-manipulation або IDOR;
- Insecure Deserialization: експлуатація неконтрольованого розпакування об’єктів;
- Insecure Configuration: небезпечні дефолтні налаштування серверів, увімкнені debug-режими, відкриті адміністративні інтерфейси;
- Вразливості API: недоліки в механізмах авторизації/аутентифікації REST/GraphQL-запитів, витік даних [5, 7].

Окрім технічних атак, останнім часом дедалі більшого поширення набувають методи соціальної інженерії — фішинг, вішинг та подібні прийоми, коли зловмисник отримує потрібну інформацію не шляхом зламу, а через обман самого користувача [8]. Часто саме так вдається дістати облікові дані або доступ до внутрішніх систем — особливо коли працівники не проходили належного навчання з кібергігієни.

Ще один важливий виклик — автоматизовані атаки. Для цього використовують боти, скрипти або фреймворки типу Selenium. Такі засоби дозволяють швидко сканувати велику кількість цілей, перевіряти веб-додатки на стандартні вразливості та навіть намагатися їх експлуатувати — без втручання людини [6, 9, 10]. У результаті це стає реальною загрозою навіть для невеликих сайтів, які раніше вважались нецікавими для атак.

Таким чином, оцінка загроз веб-додаткам є фундаментальним етапом побудови системи захисту. Вона дозволяє розробити карту ризиків, визначити найбільш імовірні сценарії атак і обґрунтовано підійти до вибору контрзаходів. Для ефективного захисту веб-додатків необхідно поєднувати технічні засоби (WAF, шифрування, контроль доступу) з організаційними заходами (навчання персоналу, аудит безпеки, оновлення ПЗ) [1].

1.2 Аналіз вразливостей OWASP Top 10

OWASP Top 10 — це загальновизнаний міжнародний список найбільш критичних вразливостей веб-додатків, який формується і періодично оновлюється фахівцями з кібербезпеки з усього світу [11]. Основна мета цієї ініціативи — донести до розробників, тестувальників і компаній інформацію про найбільш поширені загрози, аби вони могли враховувати їх у своїй роботі й робити веб-системи безпечнішими [5].

Це не просто теоретичний перелік, а реальна підбірка проблем, які найчастіше зустрічаються на практиці. Остання версія OWASP Top 10 вийшла у 2021 році й включає десять основних категорій вразливостей, що найчастіше

стають причиною зламів веб-додатків або витоків даних [5]. Цей документ вважається базовим орієнтиром для всіх, хто працює над захистом веб-середовища — як у навчальних цілях, так і в реальних проектах.

1. Broken Access Control (Порушення контролю доступу)

Це найнебезпечніша категорія згідно з OWASP 2021. Вона виникає, коли система дозволяє доступ до ресурсів або функцій, які мають бути обмеженими. Типові приклади: обхід URL, зміна ідентифікаторів (IDOR), зміна ролей, доступ до чужих даних. Згідно з OWASP, майже у 94% протестованих додатків виявлено проблеми контролю доступу.

2. Cryptographic Failures (Криптографічні помилки)

Ця категорія замінила попередню "Sensitive Data Exposure". Йдеться про неправильну реалізацію алгоритмів шифрування, слабку генерацію ключів, використання застарілих протоколів (наприклад, TLS 1.0), або зберігання паролів у відкритому вигляді. Наслідки — витік паролів, токенів, особистих даних.

3. Injection (Ін’єкції)

Включає SQL, NoSQL, OS Command, LDAP та інші ін’єкції. Уразливості виникають тоді, коли введені дані потрапляють у запит без належної фільтрації. Це дозволяє зловмиснику змінювати логіку виконання запиту, обходити автентифікацію, читати або змінювати бази даних. Класичний приклад — SQLi (' OR 1=1--).

4. Insecure Design (Ненадійне проектування)

Це нова категорія, яка стосується концептуального рівня. Вона охоплює недоліки у розробці систем, в яких відсутні принципи безпечного проєктування. Це, наприклад, відсутність threat modeling, ігнорування "least privilege", спрощена логіка контролю доступу, відсутність валідації прав.

5. Security Misconfiguration (Неправильна конфігурація безпеки)

Одна з найпоширеніших причин інцидентів. Виникає через залишення налаштування за замовчуванням, відкриті каталоги, доступ до консольних

інтерфейсів, увімкнені debug-режими. Часто атакуючий отримує root-доступ завдяки одному незахищеному елементу (наприклад, залишеному endpoint'у).

6. Vulnerable and Outdated Components (Уразливі та застарілі компоненти)

Використання сторонніх бібліотек або фреймворків, які мають відомі уразливості, є серйозним ризиком. Приклади: застарілі версії jQuery, Log4j, Apache Struts, OpenSSL. Часто такі компоненти не оновлюються розробниками або їх вразливості залишаються непоміченими.

7. Identification and Authentication Failures (Помилки автентифікації)

Охоплює широкий спектр проблем: використання простих паролів, відсутність обмеження на кількість спроб входу, небезпечне зберігання хешів, повторне використання сесій, відсутність 2FA. Атаки типу brute-force або credential stuffing є типовими прикладами.

8. Software and Data Integrity Failures (Порушення цілісності ПЗ та даних)

Ця категорія охоплює випадки, коли програми завантажують оновлення, виконують скрипти або інтегрують сторонні компоненти без перевірки цілісності та достовірності. Типовий приклад — відсутність цифрового підпису пакета оновлень, що дозволяє впровадити зловмисний код.

9. Security Logging and Monitoring Failures (Проблеми журналювання та моніторингу)

Відсутність логування або системи моніторингу призводить до того, що атаки залишаються непоміченими. Це унеможливлює своєчасне реагування на інциденти, знижує ефективність розслідувань та аудитів.

10. Server-Side Request Forgery (SSRF)

Ця вразливість дозволяє зловмиснику змусити сервер робити запити до довільних ресурсів, включаючи внутрішні. SSRF може бути використана для сканування внутрішньої інфраструктури, отримання метаданих хмари, обхід міжмережевих екранів.

Окремо варто виділити такі приклади:

- XSS (Cross-Site Scripting) — дозволяє виконати JavaScript у контексті жертви.

- CSRF (Cross-Site Request Forgery) — змушує браузер користувача виконати запит без його згоди.
- IDOR (Insecure Direct Object Reference) — дозволяє змінювати чужі ресурси (наприклад, user?id=123 → user?id=124).
- XXE (XML External Entity) — атака через обробку XML-файлів, що дає змогу читати файли або робити SSRF.
- RCE (Remote Code Execution) — критична уразливість, що дозволяє запускати довільний код на сервері.

Ці категорії часто поєднуються між собою, посилюючи вплив. Наприклад, ненадійна конфігурація (5) може відкрити вразливості для SSRF (10), а використання застарілих компонентів (6) призводить до RCE або XXE [5, 9].

Таким чином, знання OWASP Top 10 дозволяє не лише ефективно виявляти уразливості, а й впроваджувати політику безпечної розробки, проводити аудит захисту, формувати вимоги до ПЗ та підвищувати обізнаність розробників і адміністраторів щодо сучасних загроз [12].

1.3 Методи атак на веб-додатки через HTTP, MITM і сесійні механізми

Сучасні веб-додатки працюють завдяки постійній взаємодії між клієнтською частиною (тобто браузером користувача) і сервером, причому основними протоколами обміну є HTTP або HTTPS. Проблема в тому, що HTTP — це текстовий протокол, і якщо не використовується шифрування (тобто HTTPS), то весь трафік можна легко перехопити, змінити або навіть повторно відправити зловмисником [9, 13].

Без належного захисту така вразливість відкриває чимало можливостей для атак: від простого перехоплення логінів до повного викрадення сесії користувача. Більшість подібних атак базується саме на роботі з HTTP-запитами — коли зловмисник вміє їх підміняти, повторювати або модифікувати в реальному часі. Саме тому питання безпечного обміну даними між клієнтом і сервером залишається одним із ключових [4].

Маніпуляція HTTP-запитами (HTTP Request Manipulation)

Цей тип атак базується на активному втручанні в структуру та вміст HTTP-запитів, які надсилаються клієнтом до сервера. Зловмисник може змінити запит ще до його обробки сервером, використовуючи інструменти перехоплення та редагування.

Основні методи:

- Модифікація заголовків (headers): підміна User-Agent, Referer, Host може вплинути на логіку авторизації або обхід фільтрів.
- Підміна HTTP-методу: наприклад, зміна GET на POST або PUT для виконання несанкціонованих дій.
- Редагування параметрів URL чи форми: дозволяє здійснювати SQL-ін’єкції, атаки типу IDOR, або викликати сторонні API з довільними параметрами.
- Маніпуляції з cookies: зміна ідентифікатора сесії або прапорців безпеки (HttpOnly, Secure) може призвести до захоплення сесії.

Найпоширенішими інструментами для проведення таких атак є Burp Suite, OWASP ZAP, Fiddler, які дозволяють у реальному часі перехоплювати, змінювати та повторно надсилати запити [10, 14, 15]. Це відкриває широкий спектр можливостей як для етичного тестування, так і для зловмисної діяльності.

MITM (Man-in-the-Middle, атака "людина посередині")

Це один із найнебезпечніших способів атаки, при якому зловмисник перехоплює трафік між клієнтом і сервером, залишаючись непоміченим. Такі атаки є особливо ефективними в публічних мережах Wi-Fi, де відсутній контроль доступу [8].

Можливості зловмисника під час MITM:

- Перегляд конфіденційних даних: логінів, паролів, банківської інформації, токенів автентифікації.
- Підміна вмісту відповідей сервера: наприклад, впровадження шкідливого коду в HTML-сторінку.

- Ін’єкція JavaScript-коду: може призвести до XSS-атак або крадіжки cookie.
- Модифікація запитів: дозволяє вплинути на логіку запиту, змінити параметри або викликати додаткові дії.

Ефективний захист включає:

- Використання HTTPS із сертифікатом, виданим надійним центром сертифікації.
- Впровадження HTTP Strict Transport Security (HSTS).
- Шифрування даних на рівні застосунку.
- Перевірку автентичності сертифікатів.
- Використання VPN у незахищених мережах [13].

Сесійні атаки (Session Attacks)

Ці атаки орієнтовані на захоплення, підміну або повторне використання ідентифікаторів сесій користувачів. Сесія — це унікальний зв’язок між клієнтом і сервером, який зберігає стан авторизації та дозволяє користувачеві взаємодіяти з додатком без повторної автентифікації.

Основні типи сесійних атак:

- Session Hijacking: перехоплення активної сесії, часто за допомогою MITM або XSS.
- Session Fixation: нав’язування заздалегідь відомого сесійного ідентифікатора перед входом користувача.
- Session Replay: повторне використання раніше перехопленого запиту без змін.
- Predictable Session ID: якщо токен сесії легко передбачити, зловмисник може його згенерувати та використати.

Методи захисту:

- Використання криптографічно стійких токенів, які важко підібрати.
- Часове обмеження тривалості сесії та автоматичне завершення при бездіяльності.
- Прив’язка сесії до IP-адреси або агента користувача.

- Зміна токена після логіна (session regeneration).
- Інвалідація сесії при виході, зміні паролю або інших критичних діях [16].

Загальна оцінка

Методи атак, пов’язані з HTTP-протоколом і сесіями, є фундаментальними через їхню універсальність і низький поріг входу для зловмисників. Їх застосовують як у простих скрипт-кіді-атаках, так і у складних багатоетапних сценаріях кіберрозвідки. У зв’язку з цим, ефективна реалізація захисту на рівні HTTP, сесійного менеджменту та транспортного шару є обов’язковою умовою створення безпечних веб-додатків.

1.4 Аналіз реальних кіберінцидентів, пов’язаних із веб-додатками

У сучасному цифровому середовищі атаки на веб-додатки стали не лише повсякденним явищем, а й системною проблемою, яка охоплює компанії, державні структури, фінансові установи, медичні сервіси та навіть постачальників програмного забезпечення. Причини такої ситуації криються у швидкому розвитку веб-технологій, широкому використанні сторонніх компонентів, недбалості розробників та відсутності системної політики безпеки. Розповсюдженість DevOps-підходу без належної інтеграції безпекових практик лише поглиблює проблему, дозволяючи вразливостям проникати в продакшн на ранніх етапах розробки. Реальні приклади інцидентів підтверджують, що навіть великі корпорації із значними бюджетами на кібербезпеку не застраховані від критичних вразливостей, особливо якщо йдеться про людський фактор або нехтування оновленнями [1].

Одним із найбільш масштабних інцидентів у цій сфері стала атака на Equifax у 2017 році, внаслідок якої витікли персональні дані понад 147 мільйонів громадян США. Атака стала можливою через уразливість у компоненті Apache Struts, яка дозволяла віддалене виконання коду за допомогою спеціально сформованого заголовка Content-Type. Патч для цієї уразливості був доступний

ще за два місяці до атаки, однак компанія не встигла або проігнорувала його встановлення. Це дозволило атакуючим проникнути до внутрішніх систем і отримати доступ до баз даних із конфіденційною інформацією, включаючи номери соціального страхування, дати народження та адреси. Крім втрати довіри з боку клієнтів, компанія зазнала значних фінансових збитків: виплати компенсацій, штрафи регуляторів і витрати на усунення наслідків інциденту сягнули понад 700 мільйонів доларів [5].

Іншим прикладом є злом Facebook у 2018 році. Унаслідок помилки в реалізації функції «View As», яка дозволяла переглядати профіль з точки зору іншого користувача, зловмисники змогли генерувати access-токени без проходження авторизації. Ці токени фактично надавали повний доступ до облікових записів, що дозволило атакуючим отримати контроль над близько 50 мільйонами акаунтів. Комбінація помилок у механізмах авторизації та сесійного менеджменту свідчить про відсутність системного тестування безпеки перед запуском нових функцій. Після інциденту компанія примусово завершила сесії майже 90 мільйонів користувачів, однак завданої шкоди це вже не скасувало.

У тому ж році British Airways зазнала атаки, внаслідок якої було викрадено платіжні дані понад 380 000 клієнтів. Група зловмисників, що входить до угруповання Magecart, змогла впровадити шкідливий JavaScript-код у сторінку оплати. Цей скрипт перехоплював дані банківських карт у реальному часі та передавав їх на зовнішній сервер. Компанія не виявила втручання одразу, оскільки не використовувала достатньо жорстких політик Content Security Policy (CSP) та не обмежувала джерела виконання сторонніх скриптів. Внаслідок порушення вимог GDPR компанія отримала один із найбільших штрафів у розмірі 20 мільйонів фунтів.

Складніший приклад, який ілюструє не лише технічну, а й інфраструктурну вразливість, трапився у 2019 році з Capital One. Зловмисниця скористалася SSRF-вразливістю в поєднанні з неправильною конфігурацією хмарного WAF на AWS, що дозволило їй отримати доступ до метаданих EC2-інстансу. Завдяки цьому вона змогла отримати тимчасові ключі доступу та

завантажити конфіденційні дані понад 100 мільйонів користувачів, включаючи соціальні й фінансові дані [5]. Цей випадок вкотре підкреслив, наскільки важливо правильно налаштовувати хмарні інфраструктури, використовувати ізольовані ролі доступу та обмеження на рівні ресурсів.

Окремо варто згадати постійні атаки на популярну CMS WordPress, яка використовується на мільйонах сайтів по всьому світу. Зловмисники регулярно знаходять вразливості у плагінах, зокрема в одному з випадків із плагіном File Manager у 2025 році була виявлена критична уразливість, що дозволяла без автентифікації завантажити довільний PHP-файл. Це відкрило прямий шлях до повного контролю над сервером. Через масштабність інциденту тисячі сайтів були скомпрометовані. У більшості випадків винна не сама платформа, а ігнорування оновлень і перевірки плагінів перед встановленням.

Нарешті, приклад наймасштабнішої атаки на ланцюг постачання стався з компанією SolarWinds у 2025 році. Зловмисники отримали доступ до внутрішніх серверів компанії, модифікували легітимний код оновлення програмного забезпечення Orion і впровадили бекдор, який було поширено серед клієнтів через веб-інтерфейс оновлення. Серед постраждалих були державні установи США, великі IT-компанії та критична інфраструктура. Атака залишалася непоміченою понад пів року і продемонструвала вразливість процесу CI/CD без відповідного контролю цілісності, цифрових підписів і багаторівневих перевірок [17].

Ці приклади свідчать, що джерелом небезпеки є не лише технічні недоліки, але й організаційна неуважність, брак процесів моніторингу, аналізу логів та аудиту. Переважну більшість інцидентів можна було запобігти завдяки регулярному оновленню компонентів, застосуванню принципів secure by design, використанню WAF, HSTS, MFA, а також впровадженню систем виявлення атак. Реальні інциденти є неоціненним джерелом уроків, на основі яких можна створювати ефективні моделі загроз, практичні політики безпеки та сценарії реагування [1].

Висновки за розділом 1

У першому розділі було здійснено комплексний аналіз загроз, вразливостей і методів атак, що є характерними для сучасних веб-додатків. Проведене дослідження показало, що веб-додатки залишаються однією з основних цілей зловмисників через широкий спектр потенційних векторів атак, відкритість до зовнішніх підключень і часту недосконаліть у реалізації захисту.

Загрози веб-додатків класифікуються за походженням, характером дій зловмисника та спрямованістю на порушення конфіденційності, цілісності або доступності даних. Було виявлено, що найбільш поширеними є атаки, що використовують вразливості рівня введення даних, а також неправильні конфігурації систем, слабкий контроль доступу та відсутність безпечного проектування.

Окрему увагу було приділено вразливостям, включеним до OWASP Top 10 (редакції 2021 року). Цей перелік охоплює як класичні проблеми, на кшталт ін’єкцій чи слабкої автентифікації, так і новіші виклики — наприклад, ненадійне проєктування, відсутність перевірки цілісності компонентів, SSRF та проблеми з моніторингом. Аналіз показав, що більшість таких вразливостей мають спільні причини: відсутність належної валідації, використання застарілих бібліотек, недоопрацьована архітектура або нехтування захистом трафіку.

Серед розглянутих методів атак окремо варто виділити маніпуляції з HTTP-запитами, MITM-атаки (тобто "людина посередині") та різні техніки захоплення сесій. Ці методи активно використовують особливості самого протоколу HTTP, і, коли додається ще й людський фактор, вони можуть бути особливо ефективними.

Окрім цього, у розділі були розглянуті й кілька реальних кіберінцидентів останніх років — зокрема атаки на Equifax, Facebook, British Airways, Capital One, SolarWinds та випадки масового зламу через вразливості в плагінах WordPress. Аналіз показав, що в більшості цих ситуацій основною проблемою була не стільки складність атак, скільки проігноровані оновлення, відсутність належного

аудиту безпеки, використання вразливої архітектури або стороннього коду без перевірки.

Таким чином, проведений аналіз у межах першого розділу дозволяє сформувати теоретичне підґрунтя для практичної реалізації механізмів виявлення, аналізу та усунення вразливостей веб-додатків, що розглядатимуться в наступних розділах кваліфікаційної роботи. Результати відповідають поставленим у вступі завданням і наближають до досягнення загальної мети дослідження.

РОЗДІЛ 2
ІНСТРУМЕНТАРІЙ ТА ПІДХОДИ ДО ВИЯВЛЕННЯ ВРАЗЛИВОСТЕЙ У СЕРЕДОВИЩІ ВЕБ-ДОДАТКІВ

2.1 Інструменти автоматизованого тестування безпеки веб-додатків

Щоб підтримувати належний рівень безпеки веб-додатків, потрібен постійний контроль — як за самим кодом, так і за конфігураціями, логікою роботи та взаємодією з користувачем. Усе це досить складно виконувати вручну, особливо коли проєкт активно розвивається. Саме тому автоматизоване тестування безпеки набуває все більшого значення — особливо в умовах використання DevSecOps-підходу та швидких циклів розгортання оновлень [17].

Автоматизація дозволяє суттєво зекономити час на пошук критичних вразливостей, зменшити ризик людських помилок і зробити тестування безпеки частиною звичайного процесу CI/CD [8]. Тобто тести запускаються не раз на квартал, а постійно — при кожному новому коміті або релізі. Такий підхід робить безпеку невід'ємною частиною розробки, а не окремим етапом десь наприкінці.

Інструменти, які застосовують для автоматизованого тестування безпеки, можна поділити на кілька груп — залежно від того, який саме підхід вони використовують для аналізу веб-додатка. Кожен тип має свої плюси та недоліки, і вибір залежить як від цілей перевірки, так і від архітектури самого додатка.

1. SAST (Static Application Security Testing) — статичне тестування, яке виконується без запуску додатку. Здійснюється аналіз вихідного коду, байт-коду або бінарних файлів із метою виявлення потенційно небезпечних конструкцій. Такий підхід дозволяє зафіксувати SQL-ін'єкції, міжсайтове скриптування (XSS), використання застарілих API, небезпечну обробку введення або порушення логіки авторизації ще до розгортання додатку.

До найпопулярніших інструментів цієї категорії належать [4]:

- SonarQube — із широкими можливостями для інтеграції та кастомізації правил перевірки;
- Fortify Static Code Analyzer — розроблений компанією Micro Focus для великих корпоративних проєктів;
- Checkmarx — з потужною підтримкою різних мов програмування;
- Semgrep — легкий інструмент з відкритим кодом, орієнтований на швидке статичне сканування.

2. DAST (Dynamic Application Security Testing) — динамічне тестування працюючого додатку без доступу до вихідного коду. Інструмент імітує реальні атаки, взаємодіючи з інтерфейсом додатку як звичайний користувач. Це дозволяє виявити вразливості, пов’язані з поведінкою додатку, його реакцією на введення даних, а також з некоректною обробкою запитів.

До ефективних DAST-рішень належать:

- OWASP ZAP — безкоштовний open-source інструмент із активною підтримкою спільноти;
- Burp Suite Professional — з можливістю як ручного, так і автоматичного сканування;
- Acunetix — комерційне рішення з широкою підтримкою типових атак;
- Netsparker (Invicti) — з функцією верифікації вразливостей;
- AppSpider — добре підходить для великих динамічних додатків.

3. IAST (Interactive Application Security Testing) — інтерактивне тестування, яке працює усередині додатку під час його виконання, дозволяючи одночасно аналізувати як код, так і HTTP-трафік. Інструменти IAST надають більш точну інформацію про контекст вразливості, враховуючи внутрішню логіку, права доступу та змінні середовища.

Прикладами таких рішень є [8]:

- Contrast Security — підтримує аналіз у режимі реального часу;
- Seeker (Synopsys) — надає повний трасувальний шлях уразливості;
- Imperva RASP — із додатковими функціями активного блокування атак.

4. SCA (Software Composition Analysis) — аналіз складу додатку з акцентом на відкриті компоненти (open-source), бібліотеки та зовнішні залежності. Оскільки більшість сучасних додатків базуються на сторонніх модулях, безпека всього проєкту залежить від їх актуальності й надійності.

Поширені рішення для SCA:

- Snyk — швидке виявлення вразливих залежностей;
- OWASP Dependency-Check — перевірка залежностей на відповідність відомим CVE [5];
- WhiteSource Bolt — інтеграція з GitHub і Azure DevOps;
- GitHub Dependabot — автоматичне оновлення бібліотек із відомими уразливостями.

Серед переваг автоматизованих засобів тестування варто зазначити :

- високу швидкість обробки великих обсягів коду;
- можливість регулярного сканування на кожному етапі життєвого циклу ПЗ;
- інтеграцію в середовище CI/CD для забезпечення безперервного аналізу;
- зниження ймовірності пропуску типових вразливостей [16, 18].

Однак автоматизоване тестування не може повністю замінити ручну перевірку. Інструменти можуть генерувати хибні спрацьовування, не враховувати бізнес-логіку чи специфічні особливості додатку. Саме тому найбільш ефективним підходом до тестування безпеки вважається комбіноване використання автоматизованих і ручних методів, що дозволяє досягти як глибини, так і ширини виявлення ризиків [9].

2.2 Методи пентестингу веб-додатків

Пентестинг (penetration testing) — це контрольований процес моделювання атак на веб-додатки з метою виявлення вразливостей, перевірки ефективності механізмів захисту й оцінки рівня стійкості додатку до реальних кіберзагроз [17].

На відміну від автоматизованого сканування, яке зазвичай спрямоване на виявлення відомих типових помилок, пентестинг дозволяє виявляти як технічні, так і логічні вразливості, які можуть бути специфічними для конкретної реалізації веб-додатку.

Процес пентестингу охоплює кілька основних етапів. Перший — рекогносцировка (або збір інформації), під час якої пентестер аналізує загальнодоступні або частково приховані дані про веб-додаток. Збирається інформація про IP-адреси, доменні імена, DNS-записи, сервіси, порти, використовувані CMS, структуру JavaScript-файлів, URL-шляхи, технології на фронтенді та бекенді, зовнішні залежності, публічні API та інші аспекти, що можуть надати вектори атаки [6, 8]. Також можуть бути задіяні пошукові сервіси типу Shodan або Google dorking.

Другим кроком є створення карти додатку, або так зване content discovery, що включає виявлення всіх наявних сторінок, форм, параметрів, точок введення даних, API-ендпоінтів, маршрутів доступу. Це дозволяє пентестеру побудувати логічну модель взаємодії клієнта із сервером та зафіксувати усі точки, через які може відбуватись вплив на додаток.

Далі виконується аналіз параметрів та тестування на типові уразливості. Перевірці підлягають усі вхідні точки введення даних: параметри URL, поля форм, cookie, заголовки HTTP-запитів, тіло POST-запитів, параметри JSON та XML, WebSocket-повідомлення. У цьому контексті застосовуються техніки fuzzing, маніпуляція типами даних, виявлення ін’єкцій (SQLi, XSS, OS Command Injection, XXE), підміна токенів, обхід валідації на клієнтському боці, перевірка захисту від CSRF та багато іншого. Під час тестування широко використовуються інструменти на кшталт Burp Suite, OWASP ZAP, wfuzz, ffuf, sqlmap, Postman, а також власні скрипти пентестера.

Особливу увагу приділяють перевірці механізмів автентифікації та авторизації. Аналізуються можливості brute-force атак, вразливості в логіці логіна (наприклад, можливість обійти автентифікацію за допомогою параметрів у URL), вразливості типу IDOR (Insecure Direct Object References), слабкі токени,

вразливості керування сесіями, неправильне розмежування прав доступу (наприклад, звичайний користувач може виконувати дії адміністратора).

Після виявлення потенційних вразливостей переходять до етапу експлуатації, на якому виконується спроба підтвердити уразливість [9]. Метою є досягнення конкретного результату: викрадення облікових даних, обхід обмежень, виконання коду на сервері, ін'єкція JavaScript у браузер жертви, витік інформації з бази даних тощо. Це дозволяє оцінити реальні наслідки виявленої проблеми та вплив на безпеку всього додатку.

У деяких випадках, особливо при глибокому аналізі, виконується постексплуатація — аналіз можливостей закріплення в системі, ескалації привілеїв, доступу до внутрішніх сервісів компанії, lateral movement у внутрішній інфраструктурі. Такі дії є особливо актуальними для тестування додатків, пов'язаних із корпоративними середовищами або внутрішніми порталами.

Завершується пентестинг етапом звітності, де формується структурований звіт про виявлені вразливості. Звіт включає опис кожної знахідки, приклади експлуатації, HTTP-запити та відповіді, скріншоти (де потрібно), оцінку ризику (на основі CVSS або власної шкали), а також конкретні рекомендації щодо усунення кожної вразливості. Після впровадження виправлень рекомендується провести повторне тестування, щоб переконатися в ефективності заходів і виключити повторну появу тих самих дефектів. Це дозволяє замовнику пентесту або команді розробки оперативно впровадити зміни й підвищити рівень безпеки.

Методологія проведення пентестингу може варіюватись залежно від обсягу доступу до інформації. При black-box підході тестувальник не має жодної внутрішньої інформації — перевірка імітує типову поведінку стороннього зловмисника. У разі gray-box підходу пентестеру надається часткова інформація, наприклад, обліковий запис користувача або архітектурна схема додатку. У моделі white-box виконавець має повний доступ до коду, конфігурацій, документації, що дозволяє глибше дослідити складні логічні помилки та архітектурні дефекти.

Таким чином, пентестинг є незамінним етапом забезпечення безпеки веб-додатків, оскільки дозволяє моделювати реальні сценарії атак і виявляти критичні недоліки, які можуть залишитися непоміченими при автоматизованому аналізі. Його проведення має ґрунтуватися на чітко визначеній методології, професійних навичках і розумінні логіки роботи веб-додатків.

2.3 Огляд Burp Suite як потужного інструменту для тестування безпеки

Burp Suite — це один із найпопулярніших і функціонально насичених інструментів для тестування безпеки веб-додатків [10, 14]. Розроблений британською компанією PortSwigger, він поєднує в собі інструменти як для ручного, так і для автоматизованого аналізу безпеки, що робить його універсальним рішенням для фахівців з кібербезпеки.

Burp Suite працює як локальний проксі-сервер, що дозволяє перехоплювати, переглядати й змінювати HTTP(S)-трафік між браузером та веб-додатком. Це забезпечує повний контроль над усією комунікацією, даючи змогу детально аналізувати поведінку системи в реальному часі. Зокрема, пентестер має змогу вивчати заголовки, параметри, тіла запитів і відповідей, змінювати їх вручну або програмно та повторно надсилати для оцінки реакції серверу.

Архітектура Burp Suite побудована за модульним принципом. До основних компонентів належать:

- Proxy — базовий інструмент, що забезпечує перехоплення трафіку між клієнтом і сервером. Можна налаштовувати правила фільтрації, декодувати закодовані дані, автоматично змінювати запити, додавати breakpoints.
- Target — модуль, який створює карту структури веб-додатку на основі зібраного трафіку. Дає змогу виявляти зони високого ризику та ефективно організовувати подальше тестування.

- Intruder — потужний фреймворк для виконання атак типу fuzzing, brute-force, enumeration. Підтримує шаблони введення, payload-генератори, умовні дії на основі відповідей.
- Repeater — дає змогу вручну повторювати запити з неодноразовими змінами для глибокого аналізу конкретної точки взаємодії.
- Scanner (доступний лише в Professional-версії) — автоматизований сканер, який використовує десятки технік для виявлення типових уразливостей, таких як SQLi, XSS, CSRF, SSTI, SSRF, IDOR та інші.
- Sequencer — аналізує випадковість і ентропію токенів сесій, ідентифікаторів, анти-CSRF токенів тощо.
- Decoder та Comparer — допоміжні утиліти для кодування/декодування даних (Base64, URL, hex) та порівняння різних версій запитів/відповідей.

Однією з ключових переваг Burp Suite є можливість розширення. Інструмент підтримує розробку та встановлення власних розширень за допомогою API на мовах Java, Python (через Jython) та JavaScript. Крім того, існує Burp BApp Store — онлайн-репозиторій плагінів, створених спільнотою, що включає рішення для аналізу Open Redirect, DOM-based XSS, JSON Web Token (JWT), SSRF, автоматичної обробки CAPTCHA, перехоплення WebSocket-трафіку тощо [18].

Burp Suite постачається у двох основних версіях:

- Community Edition — безкоштовна, з обмеженим функціоналом. Підходить для базового аналізу, навчання, ознайомлення з інтерфейсом.
- Professional Edition — платна версія, яка включає повноцінний сканер, розширені функції Intruder, інтеграцію з CI/CD, створення кастомних профілів тестування, звітність у форматах HTML/PDF/XML.

Завдяки широкому функціоналу, зручному інтерфейсу, активній спільноті та гнучкості налаштувань, Burp Suite зарекомендував себе як де-факто стандарт у сфері пентестингу веб-додатків. Його активно використовують як незалежні дослідники, так і професійні команди з безпеки в комерційних та державних організаціях. Програма постійно оновлюється, що дозволяє оперативно

реагувати на появу нових типів вразливостей і забезпечувати актуальність функцій.

На відміну від багатьох інших рішень, Burp Suite надає користувачеві повний контроль над процесом тестування, не обмежуючись автоматичним пошуком. Це дозволяє не лише знаходити вразливості, а й глибоко аналізувати логіку роботи веб-додатку, перевіряти складні сценарії взаємодії, виявляти неочевидні аномалії та шляхи обходу. Важливо зазначити, що Burp Suite підтримує створення та інтеграцію власних розширень через BApp Store, що дає змогу розширювати функціонал відповідно до специфічних потреб проекту.

Таким чином, Burp Suite виступає не просто інструментом сканування, а багатофункціональним середовищем для проведення повноцінного циклу безпекового тестування веб-додатків.

2.4 Основні функції Burp Suite

Burp Suite є багатофункціональним інструментом, що дозволяє ефективно проводити аудит безпеки веб-додатків. Його можливості охоплюють як ручне тестування, так і часткову або повну автоматизацію процесів. Завдяки цьому Burp Suite активно застосовується як у навчальних цілях, так і в професійній пентест-практиці [14, 15]. Найбільш ключовими функціональними можливостями, що використовуються для виявлення, аналізу та експлуатації вразливостей, є: перехоплення і модифікація HTTP-запитів, аналіз структури веб-додатку, а також тестування точок взаємодії на наявність типових загроз.

Перехоплення та модифікація HTTP-запитів є базовою функцією Burp Suite, реалізованою через компонент Proxy. Після налаштування браузера на роботу через локальний проксі-сервер Burp, усі HTTP і HTTPS-запити проходять через інтерфейс програми [9]. Користувач має змогу в реальному часі переглядати, змінювати, відкладати або навіть блокувати запити до того, як вони надійдуть на сервер. Це дозволяє вручну змінювати параметри URL, тіла POST-запитів, куки, заголовки (User-Agent, Referer тощо), симулюючи поведінку

зловмисника. Такий підхід дає змогу ефективно тестувати систему на стійкість до ін’єкцій, підробки запитів, обхід контролю доступу та інші типові вразливості.

Аналіз структури веб-додатку реалізується через вкладку Target. Burp Suite формує карту додатку, збираючи та організовуючи всю перехоплену інформацію у вигляді дерева. Відображаються усі відомі ендпоінти, методи запитів (GET, POST, PUT, DELETE тощо), параметри, куки, відповіді сервера, HTTP-коди, а також розмір і тип контенту. Це дозволяє швидко виявляти приховані функції, дублікати, точки введення, а також визначати пріоритетні зони для аналізу безпеки. Важливо, що на цьому етапі формується логічне уявлення про внутрішню архітектуру та взаємозв’язки у веб-додатку.

Виявлення та експлуатація вразливостей реалізується за допомогою таких модулів, як Intruder, Repeater та Scanner:

- Intruder використовується для автоматизованого fuzzing-тестування. Він дозволяє підставляти набір значень у параметри запитів (наприклад, імена користувачів, SQL-вирази, XSS-пейлоади) та аналізувати реакцію серверу. Intruder підтримує різні режими атаки: Sniper, Battering Ram, Pitchfork, Cluster Bomb, кожен з яких має своє призначення у конкретних сценаріях.
- Repeater застосовується для ручного тестування. Цей інструмент дозволяє багаторазово надсилати один і той самий запит із різними параметрами та порівнювати відповіді. Його використовують для перевірки логіки обробки даних, поведінки додатку при помилковому або шкідливому введенні, обходу фільтрів тощо.
- Scanner — модуль автоматичного виявлення вразливостей, доступний лише у Professional-версії Burp Suite. Він реалізує сканування за десятками правил, зокрема для OWASP Top 10: SQL-ін’єкції, XSS, CSRF, IDOR, SSRF, відкриті редиректи, уразливі заголовки, небезпечні файли тощо. Scanner оцінює ризики, групує вразливості та створює детальні звіти.

Крім основних інструментів, Burp Suite також містить:

- Decoder — для декодування або енкодування даних у форматах Base64, URL, Hex, HTML.

- Comparer — для порівняння двох запитів або відповідей, наприклад, до і після модифікацій.
- Extender — дає змогу підключати додаткові модулі з BApp Store або створювати власні плагіни на Java, Python або JavaScript через API [15, 18].

Таким чином, Burp Suite забезпечує повноцінну підтримку як для початкового збору інформації, так і для глибокого аналізу безпеки, дозволяючи працювати як з низькорівневими запитами, так і з високорівневими автоматизованими сценаріями. Його широкі функціональні можливості поєднують точність ручного тестування з ефективністю автоматизації, що робить Burp Suite універсальним інструментом у сфері веб-безпеки.

2.5 Порівняння Burp Suite з іншими інструментами тестування безпеки

Burp Suite є одним із найпоширеніших інструментів у сфері безпеки веб-додатків, однак ринок пропонує низку альтернатив, які застосовуються як окремо, так і в комбінації з ним. Для повного розуміння переваг і недоліків кожного з рішень доцільно порівняти функціональні можливості, зручність використання, точність виявлення вразливостей та можливості інтеграції в інфраструктуру розробки [19, 20].

Одним із найвідоміших безкоштовних інструментів є OWASP ZAP (Zed Attack Proxy). Цей продукт, як і Burp Suite, дозволяє перехоплювати запити, автоматично сканувати веб-додатки, використовувати плагіни та власні скрипти. Основною перевагою ZAP є його відкритий код і вільна доступність, що робить його привабливим для студентів і ентузіастів. Проте він поступається Burp Suite у зручності інтерфейсу та точності автоматичного сканування. Зокрема, Burp Scanner (у професійній версії) надає глибший та більш контекстуальний аналіз, ніж сканер ZAP [14].

Acunetix — комерційний інструмент, орієнтований на повну автоматизацію. Його основна функціональність полягає у швидкому виявленні

типових веб-вразливостей, таких як SQL-ін’єкції, XSS, CSRF, SSRF тощо. Acunetix підтримує тестування CMS-платформ (WordPress, Joomla) та забезпечує зручний вивід результатів у вигляді звітів. Його основним недоліком є обмежені можливості для ручного втручання — на відміну від Burp Suite, який дозволяє глибоку маніпуляцію HTTP-запитами і тонке налаштування [15, 18].

Інший конкурент — Netsparker (тепер відомий як Invicti) — вирізняється високою точністю, низьким рівнем хибнопозитивних результатів і глибокою інтеграцією з CI/CD-процесами. Цей інструмент оптимізовано для великих компаній і корпоративного використання. Проте, як і Acunetix, він більше підходить для автоматичного сканування, ніж для ручного аналізу, що обмежує його застосування в складних сценаріях або під час пентестів логічного рівня [20].

Для статичного аналізу коду (SAST) використовуються інші інструменти, такі як SonarQube, Checkmarx, Fortify. Вони дозволяють виявляти вразливості на рівні вихідного коду, не запускаючи веб-додаток. Це забезпечує раннє виявлення помилок на етапі розробки. Однак такі рішення не здатні перевірити поведінку програми під час виконання, що робить їх доповненням, але не альтернативою для динамічного тестування, яке реалізує Burp Suite [20].

Таким чином, на відміну від більшості конкурентів, Burp Suite поєднує часткову автоматизацію з потужними інструментами ручного тестування, що дозволяє проводити як стандартне сканування, так і глибокий аудит логіки веб-додатку [10, 21]. Його модульна архітектура, розширюваність через плагіни, велика спільнота користувачів та активна підтримка з боку розробників роблять його універсальним рішенням у сфері веб-безпеки.

Загалом, хоча існують інструменти з потужнішими автоматизованими механізмами або вузькою спеціалізацією, саме гнучкість, інтерактивність та ефективність Burp Suite забезпечують його провідні позиції серед засобів тестування безпеки веб-додатків.

Висновки за розділом 2

У другому розділі було розглянуто сучасні підходи, інструменти та методики виявлення вразливостей у веб-додатках. Особливу увагу приділено автоматизованим засобам тестування, таким як SAST, DAST, IAST та SCA, які дозволяють ефективно виявляти широкий спектр технічних недоліків у коді, конфігураціях та логіці веб-додатків. Встановлено, що найбільш повне охоплення досягається шляхом поєднання автоматизованого та ручного аналізу, оскільки автоматичні сканери не завжди здатні точно виявити складні логічні помилки або специфічні вектори атак.

Пентестинг, як практичний метод перевірки безпеки, довів свою ефективність у моделюванні реальних атак, виявленні недокументованих або складно детектованих уразливостей. Завдяки гнучкості, він дозволяє дослідити не лише технічні аспекти, а й логіку бізнес-процесів веб-додатка.

Окремо проаналізовано інструмент Burp Suite, який поєднує широкий функціонал, модульність, підтримку ручного й автоматизованого тестування та адаптивність до різних сценаріїв пентестингу. Розглянуті його можливості, ключові функції та переваги над іншими інструментами, включаючи OWASP ZAP, Acunetix, Netsparker, а також порівняння з рішеннями для статичного аналізу коду. Отримані результати підтверджують, що Burp Suite є одним із найефективніших рішень для глибокого аналізу безпеки веб-додатків у різних умовах використання.

Таким чином, сформовано теоретичну та методичну базу для переходу до практичного етапу дослідження — тестування веб-додатку в реальному середовищі із застосуванням Burp Suite.

РОЗДІЛ 3
ПРАКТИЧНА РЕАЛІЗАЦІЯ ТА ОЦІНКА ЕФЕКТИВНОСТІ ЗАХОДІВ БЕЗПЕКИ: ТЕСТУВАННЯ БЕЗПЕКИ ВЕБ-ДОДАТКА ЗА ДОПОМОГОЮ BURP SUITE

3.1 Вибір тестового середовища та налаштування Burp Suite

Для проведення практичної частини дослідження було обрано тестове середовище Damn Vulnerable Web Application (DVWA) [22, 23]. Це навчальний веб-додаток з навмисно вбудованими вразливостями, що широко використовується у галузі кібербезпеки для опанування технік ручного тестування, аналізу типових атак та вивчення способів захисту [24]. DVWA охоплює більшість категорій з OWASP Top 10, зокрема SQL-ін’єкції, XSS, CSRF, командні ін’єкції, IDOR, погане керування сесіями тощо, що робить його ідеальним прикладом для демонстрації реальних сценаріїв атак.

Підготовка тестового середовища DVWA здійснювалася поетапно:

1. Інсталяція DVWA у середовищі Kali Linux продемонстрована на (рис 3.1):

1.1. Використано віртуальну машину з попередньо встановленою Kali Linux.

1.2. За допомогою менеджера пакетів apt встановлено необхідні компоненти: веб-сервер Apache2, базу даних MariaDB, PHP та супутні розширення.

Рисунок 3.1 – Встановлення компонентів через термінал

1.3 Склоновано репозиторій DVWA з GitHub (рис 3.2):

git clone https://github.com/digininja/DVWA.git /var/www/html/dvwa

```
root@kali:~# sudo git clone https://github.com/digininja/DVWA.git /var/www/html/dvwa
Cloning into '/var/www/html/dvwa'...
remote: Enumerating objects: 5154, done.
remote: Counting objects: 100% (107/107), done.
remote: Compressing objects: 100% (33/33), done.
remote: Total 5154 (delta 88), reused 74 (delta 74), pack-reused 5047 (from 4)
Receiving objects: 100% (5154/5154), 2.52 MiB | 7.39 MiB/s, done.
Resolving deltas: 100% (2508/2508), done.
```

Рисунок 3.2 – Клонування через термінал

2. Налаштування конфігурації продемонстровано на (рисунках 3.3 – 3.6):

2.1. Відредаговано файл config.inc.php, де було вказано параметри доступу до локальної бази даних.

```
root@kali:~# cd /var/www/html/dvwa/config
root@kali:/var/www/html/dvwa/config# cp config.inc.php.dist config.inc.php
root@kali:/var/www/html/dvwa/config# sudo nano config.inc.php
root@kali:/var/www/html/dvwa/config#
```

Рисунок 3.3 – Відкриття файлу

```
 GNU nano 8.1                                config.inc.php
<?php

# If you are having problems connecting to the MySQL database and all of the variables below are c>
# try changing the 'db_server' variable from localhost to 127.0.0.1. Fixes a problem due to socket>
#   Thanks to @digininja for the fix.

# Database management system to use
$DBMS = getenv('DBMS') ?: 'MySQL';
#$DBMS = 'PGSQL'; // Currently disabled

# Database variables
#   WARNING: The database specified under db_database WILL BE ENTIRELY DELETED during setup.
#   Please use a database dedicated to DVWA.
#
# If you are using MariaDB then you cannot use root, you must use create a dedicated DVWA user.
#   See README.md for more information on this.
$_DVWA = array();
$_DVWA[ 'db_server' ]   = getenv('DB_SERVER') ?: '127.0.0.1';
$_DVWA[ 'db_database' ] = getenv('DB_DATABASE') ?: 'dvwa';
$_DVWA[ 'db_user' ]     = getenv('DB_USER') ?: 'dvwa';
$_DVWA[ 'db_password' ] = getenv('DB_PASSWORD') ?: 'p@ssw0rd';
$_DVWA[ 'db_port']      = getenv('DB_PORT') ?: '3306';

# ReCAPTCHA settings
#   Used for the 'Insecure CAPTCHA' module

^G Help          ^O Write Out     ^F Where Is      ^K Cut           ^T Execute       ^C Location
^X Exit          ^R Read File     ^\ Replace       ^U Paste         ^J Justify       ^/ Go To Line
```

Рисунок 3.4 – Редагування файлу

2.2. Увімкнено параметр allow_url_include у конфігураційному файлі PHP для коректної роботи частини вразливостей.

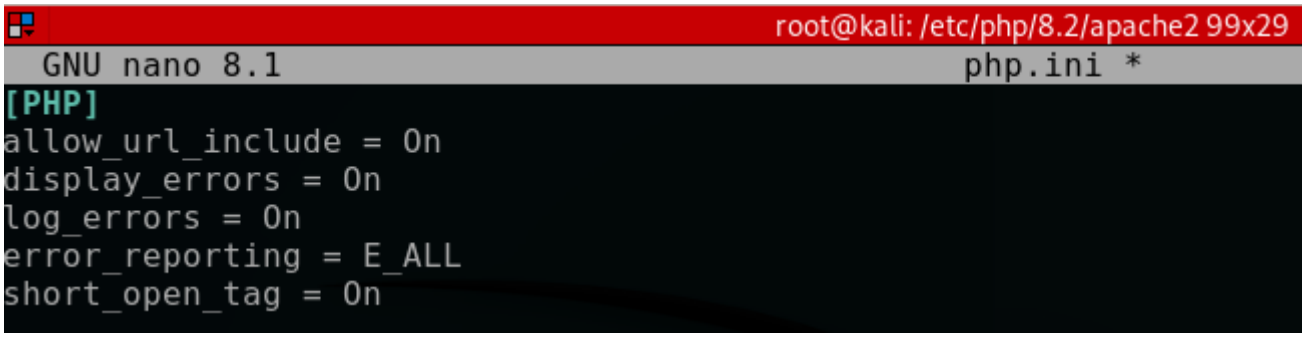

Рисунок 3.5 – Увімкнення необхідних параметрів

2.3. Перезапущено Apache і MySQL для застосування змін.

2.4. Надання прав на папку DVWA

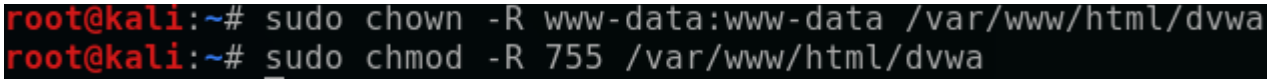

Рисунок 3.6 – Перезапуск сервісу

3. На (рисунку 3.7) продемонстровано ініціалізацію бази даних DVWA:

3.1. Перейдено у браузері за адресою http://localhost/dvwa/setup.php.

3.2. Натиснуто кнопку для створення необхідної структури бази даних у MySQL.

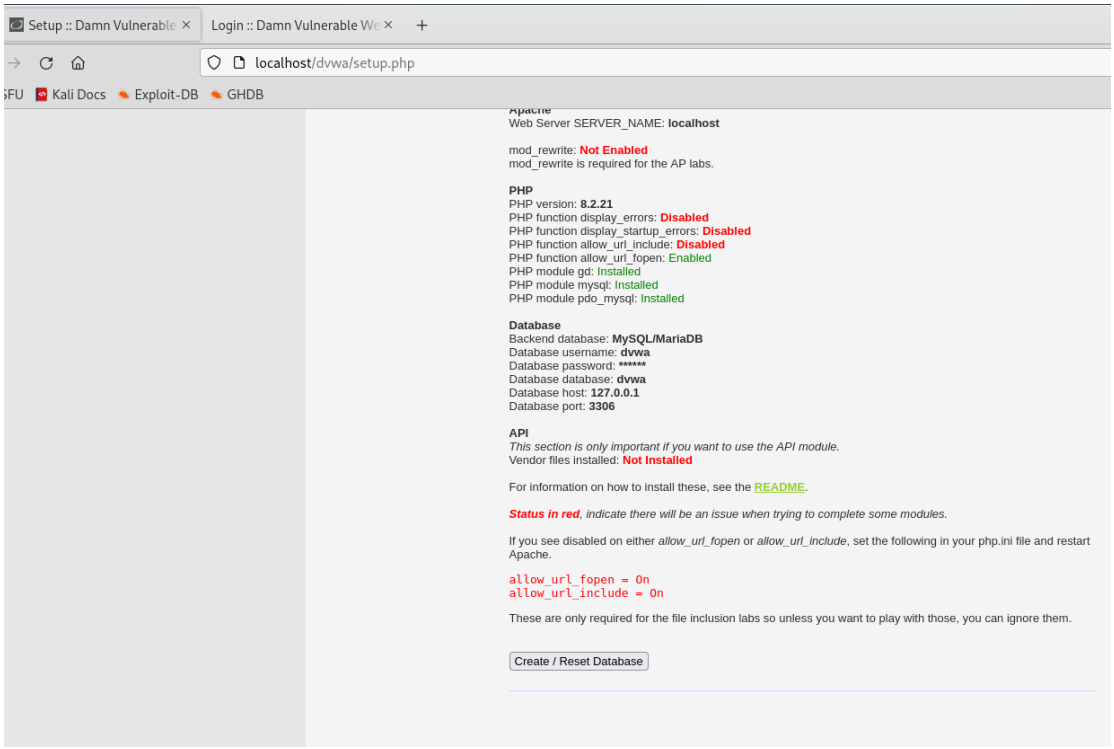

Рисунок 3.7 – Параметри і налаштування бази даних

3.3. Підтверджено, що база даних створена успішно.

```
root@kali:~# sudo service mysql start
root@kali:~# sudo mysql -u root
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 43
Server version: 11.4.2-MariaDB-4 Debian n/a

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Support MariaDB developers by giving a star at https://github.com/MariaDB/server
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]> CREATE DATABASE dvwa;
Query OK, 1 row affected (0.001 sec)

MariaDB [(none)]> CREATE USER 'dvwa'@'localhost' IDENTIFIED BY 'p@ssw0rd';
Query OK, 0 rows affected (0.005 sec)

MariaDB [(none)]> GRANT ALL PRIVILEGES ON dvwa.* TO 'dvwa'@'localhost';
Query OK, 0 rows affected (0.001 sec)

MariaDB [(none)]> FLUSH PRIVILEGES;
Query OK, 0 rows affected (0.001 sec)

MariaDB [(none)]> EXIT;
```

Рисунок 3.8 – Відповідь від БД

4. Налаштування рівня складності безпеки:

В інтерфейсі DVWA встановлено рівень безпеки "Low" (мінімальний захист) для безперешкодного тестування найпоширеніших вразливостей без впливу фільтрів, захисту або WAF.

Налаштування Burp Suite для взаємодії з DVWA передбачало:

1. Запуск Burp Suite (рис. 3.9):

1.1. Обрана версія Burp Suite Professional для повного доступу до сканера, Intruder, Repeater та інших модулів.

```
root@kali:~# burpsuite
```

Рисунок 3.9 – Команда запуску Burp Suite

2. Налаштування браузера Firefox для роботи з Burp (рис. 3.10):

2.1. У налаштуваннях проксі вказано адресу 127.0.0.1 і порт 8080.

2.2. Завантажено самопідписаний сертифікат з Burp (http://burp) та додано його до довірених у Firefox.

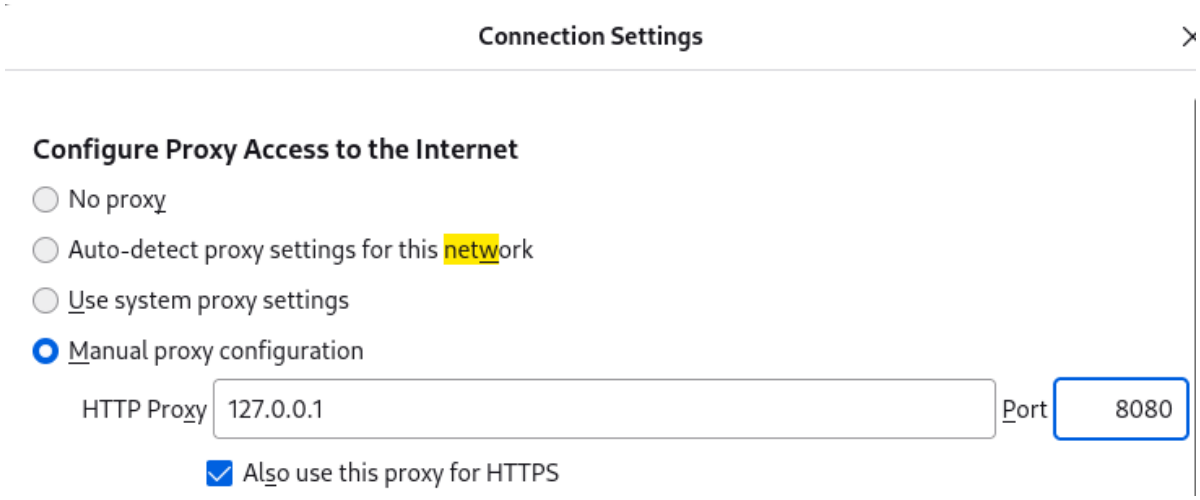

Рисунок 3.10 – Налаштування мережі

3. Перевірка перехоплення трафіку:

3.1. Увімкнено Proxy > Intercept > On.

3.2. Перехід на будь-який сайт показав успішне перехоплення HTTP(S)-трафіку, що підтвердило коректну інтеграцію браузера з Burp.

4. Визначення цільового домену DVWA:

4.1. У вкладці Target було додано URL http://localhost/dvwa до Scope, щоб обмежити аналіз лише цим додатком (рис. 3.11).

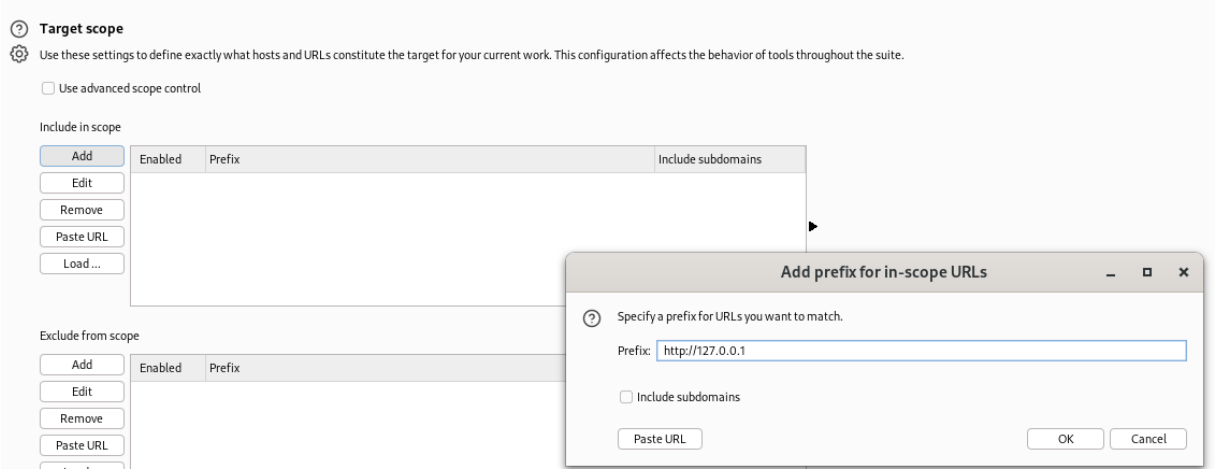

Рисунок 3.11 – Налаштування домену

4.2. Активовано автоматичне формування карти веб-додатку (Site Map) для спрощення навігації між сторінками і виявлення точок введення.

5. Активація допоміжних модулів:

5.1. Repeater — для ручної модифікації запитів.

5.2. Intruder — для тестування параметрів на стійкість до ін’єкцій та brute-force атак.

5.3. Scanner — для автоматичного виявлення вразливостей (лише у Professional версії).

5.4. Logger — для збереження всіх запитів та відповідей, що дозволяє відтворювати кроки та готувати детальний звіт.

6. Збереження проекту та сесії:

6.1. Для запобігання втраті даних налаштовано автоматичне збереження сесій Burp Suite у вигляді проектного файлу .burp.

Сформоване середовище DVWA у зв’язці з Burp Suite є надійною та гнучкою платформою для практичного тестування безпеки. Воно дає змогу відпрацьовувати методики виявлення вразливостей, досліджувати поведінку веб-додатків у різних сценаріях атак, а також розуміти, як працюють засоби захисту на практиці.

3.2 Проведення аналізу вразливостей веб-додатка

Після налаштування середовища DVWA у віртуальній машині Kali Linux та конфігурації інструменту Burp Suite було проведено систематичне тестування безпеки веб-додатку. Метою цього етапу є виявлення типових вразливостей, які найчастіше зустрічаються у практиці кібербезпеки відповідно до рейтингу OWASP Top 10. У ході роботи особливу увагу було приділено ін’єкціям, скриптовим атакам, уразливостям автентифікації, передачі даних та іншим аспектам.

1. SQL-ін’єкції (SQLi)

У модулі SQL Injection DVWA перевіряється вразливість введення користувацьких даних у SQL-запити без належної фільтрації. Було протестовано поле, у якому за умовчанням очікується числовий ідентифікатор користувача.

Підставлялися ін’єкційні конструкції, продемонстровані на (рисунках 3.12-3.14):

- ' OR 1=1 –

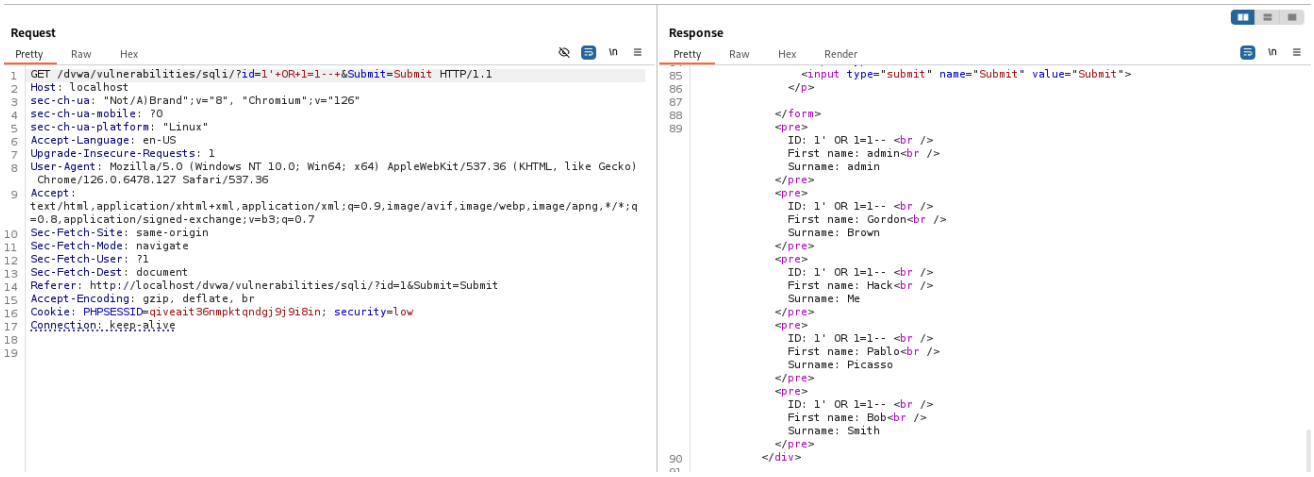

Рисунок 3.12 – Реалізація ін’єкції

- ' UNION SELECT null, version() –

Рисунок 3.13 – Реалізація ін’єкції

- ' AND (SELECT COUNT(*) FROM users) > 0 –

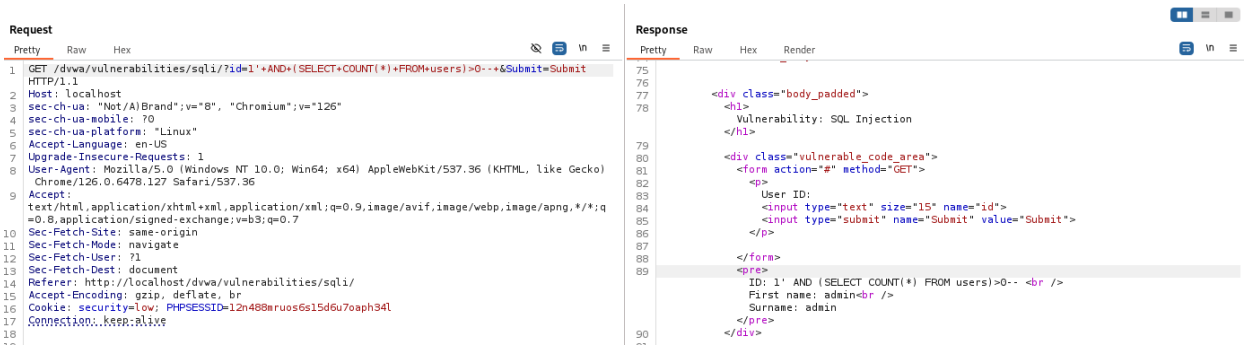

Рисунок 3.14 – Реалізація ін’єкції

У відповідь сервер повертав всі записи з бази даних, що свідчить про відсутність параметризованих запитів і вразливість до SQLi. Запити аналізувалися через Proxy та Repeater в Burp Suite. За допомогою Intruder здійснено fuzzing — автоматичну підстановку SQL-пейлоадів у параметри id.

2. Cross-Site Scripting (XSS)

У розділі XSS (Reflected) здійснювалася перевірка на виконання довільного JavaScript-коду. Використовувалися як стандартні, так і обфусковані форми скриптів:

- <script>alert("XSS")</script>

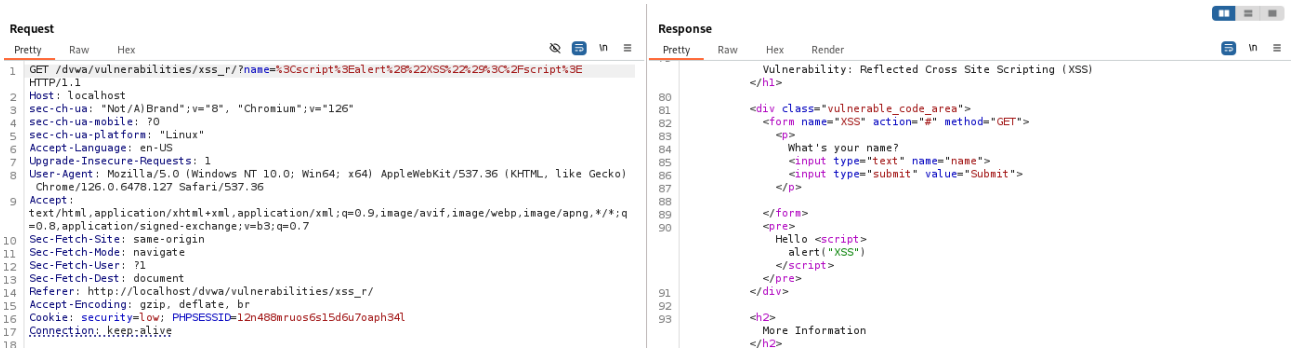

Рисунок 3.15 – Реалізація атаки

Рисунок 3.16 – Результат атаки

- <img src="x" onerror="alert('XSS')">

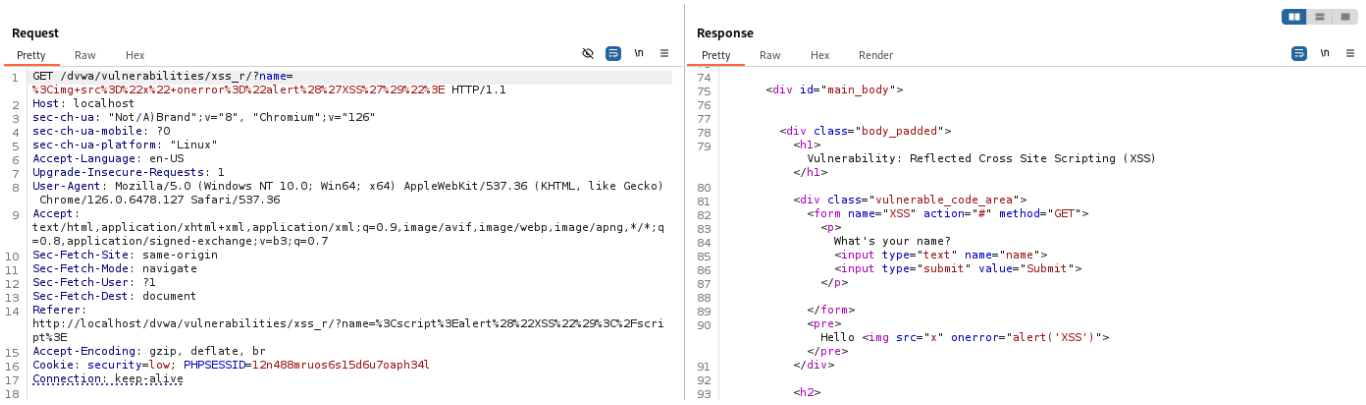

Рисунок 3.17 – Реалізація атаки

- "><svg/onload=confirm(1)>

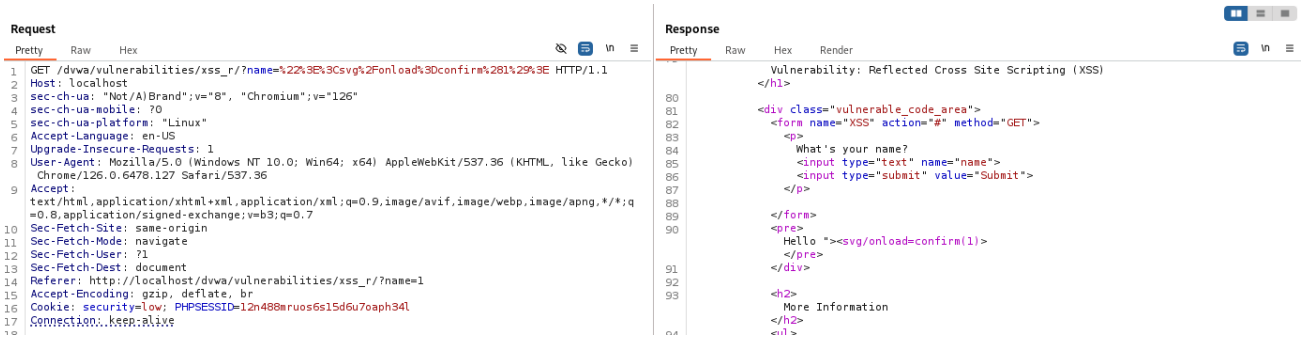

Рисунок 3.18 – Реалізація атаки

Через Repeater запити вручну змінювалися для обходу можливих фільтрів. У відповідях сервера код не був належно екранований і виконувався у браузері, що підтвердило XSS-вразливість.

3. Cross-Site Request Forgery (CSRF)

DVWA має окремий модуль CSRF, який демонструє, як можна здійснити зміну пароля користувача без його відома. Використано шаблон HTML-сторінки з прихованою формою, яка відправляє POST-запит зі зміною пароля.

Після відкриття сторінки в браузері користувача з активною сесією запит автоматично відправлявся, і пароль змінювався без взаємодії з інтерфейсом. Таким чином, відсутність CSRF-токенів підтверджена.

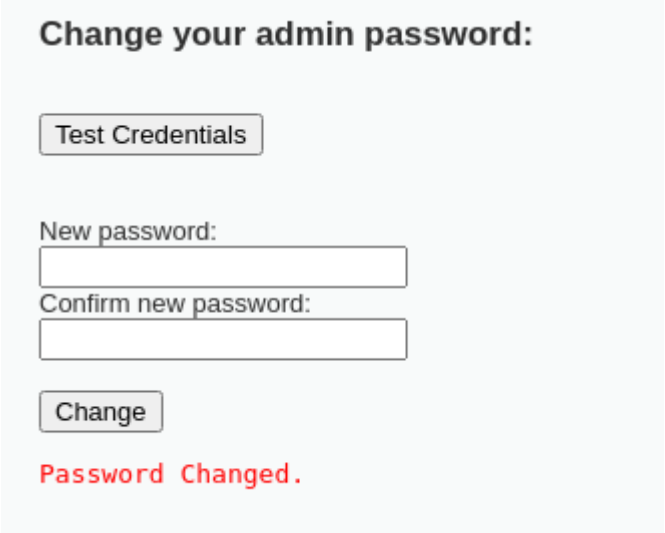

Рисунок 3.19 – Зміна паролю

Перехоплення зміни пароля через Burp:

```
GET /dvwa/vulnerabilities/csrf/?password_new=12345&password_conf=12345&Change=Change HTTP/1.1
Host: localhost
sec-ch-ua: "Not/A)Brand";v="8", "Chromium";v="126"
sec-ch-ua-mobile: ?0
sec-ch-ua-platform: "Linux"
Accept-Language: en-US
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/126.0.6478.127 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
Referer: http://localhost/dvwa/vulnerabilities/csrf/
Accept-Encoding: gzip, deflate, br
Cookie: security=low; PHPSESSID=12n488mruos6s15d6u7oaph34l
Connection: keep-alive
```

Рисунок 3.20 – Перехоплений запит

Створення простої HTML-сторінки з редіректом:

```
root@kali: ~/Desktop
root@kali: ~/Desktop 165x44
  GNU nano 8.1                                    index.html
<html>
  <body onload="window.location='http://localhost/dvwa/vulnerabilities/csrf/?password_new=hackedpass&password_conf=hackedpass&Change=Change'">
  </body>
</html>
```

Рисунок 3.21 – Код створенної сторінки

Тестування атаки:

Request to http://localhost:80 [127.0.0.1]

Forward | Drop | Intercept is on | Action | Open browser

Pretty | Raw | Hex

```
1  GET /dvwa/vulnerabilities/csrf/?password_new=hackedpass&password_conf=hackedpass&Change=Change HTTP/1.1
2  Host: localhost
3  sec-ch-ua: "Not/A)Brand";v="8", "Chromium";v="126"
4  sec-ch-ua-mobile: ?0
5  sec-ch-ua-platform: "Linux"
6  Accept-Language: en-US
7  Upgrade-Insecure-Requests: 1
8  User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/126.0.6478.127 Safari/537.36
9  Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
10 Sec-Fetch-Site: cross-site
11 Sec-Fetch-Mode: navigate
12 Sec-Fetch-Dest: document
13 Accept-Encoding: gzip, deflate, br
14 Cookie: PHPSESSID=6j9e20pvfauk15fn2s00qlj23u; security=low
15 Connection: keep-alive
16
```

Рисунок 3.22 – Реалізація атаки

Пароль автоматично змінюється на hackedpass без взаємодії. А 12345 вже не дійсний.

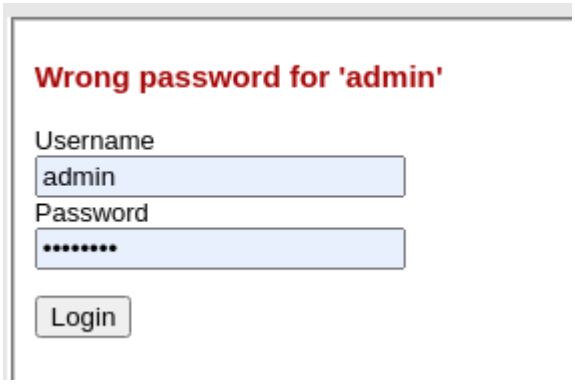

Рисунок 3.23 – Повідомлення про невірний пароль

4. Command Injection

Модуль Command Injection дозволяє ввести IP-адресу для перевірки доступності. У поле було введено команди типу:

- 127.0.0.1; whoami

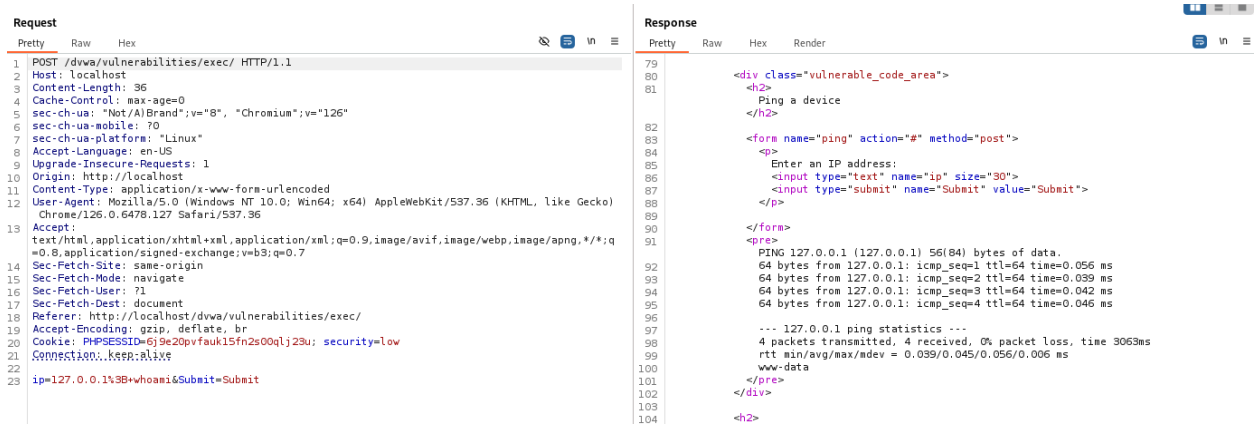

Рисунок 3.24 – Реалізація атаки

- 8.8.8.8 && cat /etc/passwd

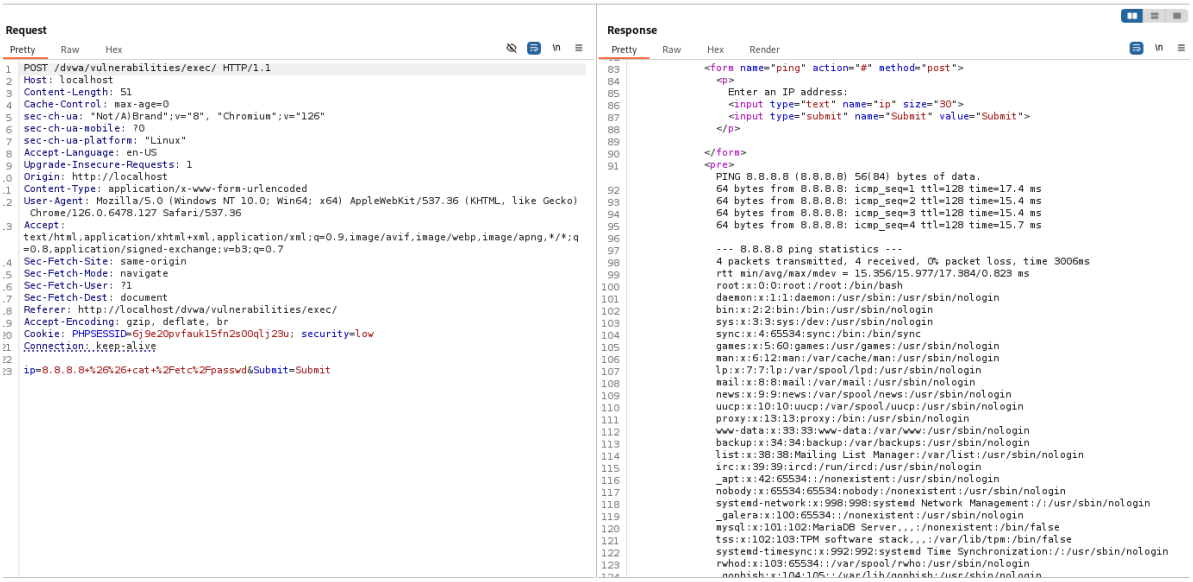

Рисунок 3.25 – Реалізація атаки

Результат повертався у відповіді сервера, що свідчить про вразливість до ін’єкції команд операційної системи. Такий підхід дозволяє не лише читати файли, а й виконувати довільні команди на сервері.

5. File Inclusion

У модулі File Inclusion тестувалося включення зовнішніх або локальних файлів. Через параметр page було спробовано підставити:

- /etc/passwd

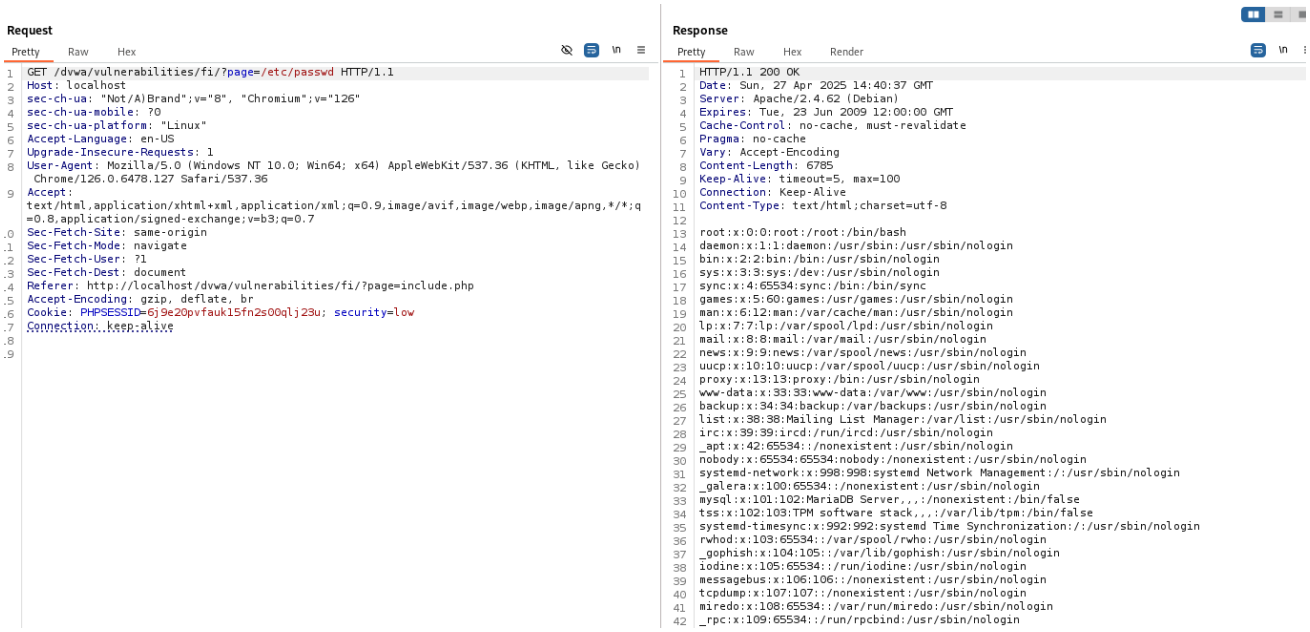

Рисунок 3.26 – Реалізація атаки і вивід отриманого тексту

DVWA дозволяв виконати включення файлів, що в реальних умовах може призвести до RCE або крадіжки облікових даних.

6. Insecure Login & Session Management

У середовищі DVWA виявлено використання простих сесій без захищених куків (відсутні флаги HttpOnly та Secure). Через Burp Suite було перехоплено сесійний cookie (PHPSESSID), який легко скопіювати до іншого браузера, що підтвердило можливість Session Hijacking.

```
GET /dvwa/index.php HTTP/1.1
Host: localhost
Cache-Control: max-age=0
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/126.0.6478.127 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
sec-ch-ua: "Not/A)Brand";v="8", "Chromium";v="126"
sec-ch-ua-mobile: ?0
sec-ch-ua-platform: "Linux"
Accept-Language: en-US
Referer: http://localhost/dvwa/login.php
Accept-Encoding: gzip, deflate, br
Cookie: PHPSESSID=6j9e20pvfauk15fn2s00qlj23u; security=low
Connection: keep-alive
```

Рисунок 3.27 – Перехоплений запит з даними

Також було виявлено відсутність обмежень на кількість спроб входу (відсутній захист від brute-force). Модуль Brute Force у DVWA дозволив за допомогою Intruder перебрати логін/пароль за словником.

Рисунок 3.28 – Вибір змінних для перебору

Вибір словника:

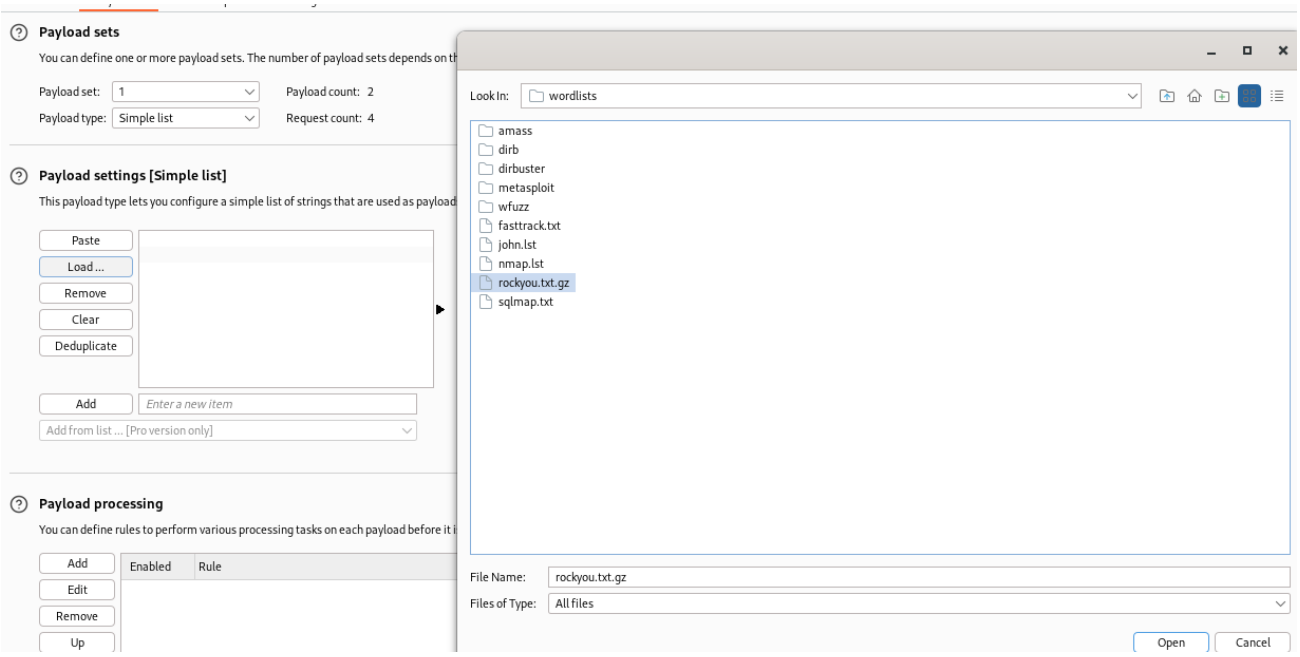

Рисунок 3.29 – Вибір словника для реалізації атаки

Intruder успішно перебрав весь словник, без жодного неопрацьованого запиту:

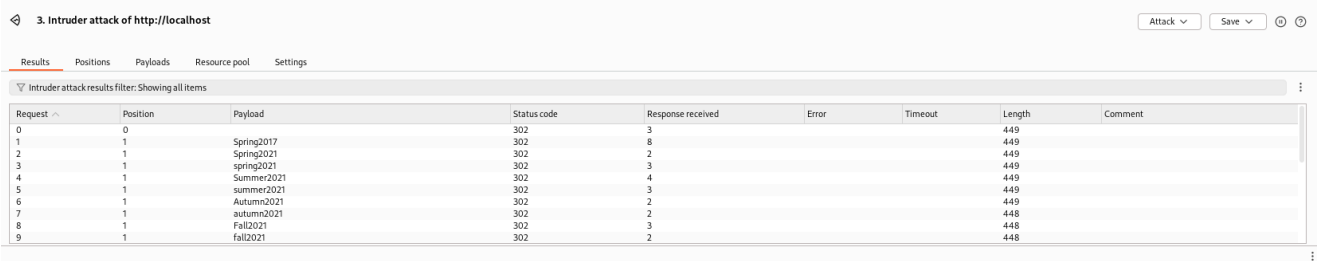

Рисунок 3.30 – Демонстрація списку спроб

7. Відсутність шифрування переданих даних (HTTP)

DVWA не використовує HTTPS, що було продемонстровано через Burp Suite: усі дані (логіни, паролі, параметри запитів) передаються у відкритому вигляді. Це створює умови для атаки типу Man-in-the-Middle (MITM) у публічних мережах.

Рисунок 3.31 – Запит без шифрування

8. Вразливості доступу (Broken Access Control, IDOR)

Було протестовано пряме звернення до URL з відповідним параметром dvwa/../.., id=2 тощо. Усі профілі були доступні незалежно від прав користувача. Відсутність перевірки прав доступу дозволяє здійснити Horizontal Privilege Escalation, що є прикладом IDOR (Insecure Direct Object References). Приклад:

Авторизація користувача без прав адміністратора:

Рисунок 3.32 – Форма авторизації

Вільний доступ до обмежених сторінок за допомогою прямого звернення до URL:

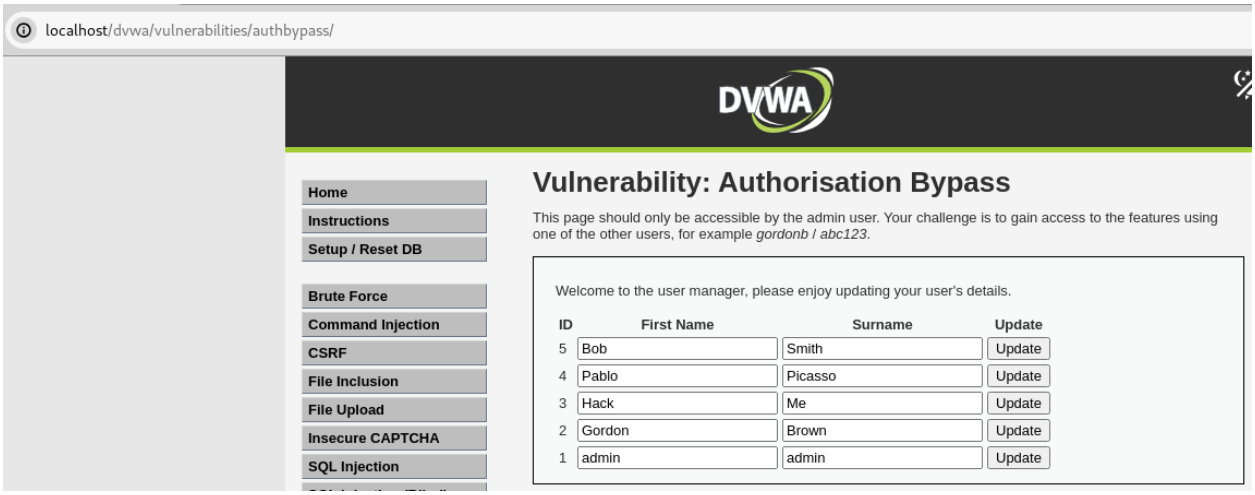

Рисунок 3.33 – Сторінка без обмежень доступу

Проміжні висновки:

Проведене тестування середовища DVWA за допомогою Burp Suite дозволило ідентифікувати низку типових та критичних вразливостей, що входять до OWASP Top 10. Всі атаки здійснювалися в контрольованому середовищі для імітації реальних кіберзагроз. Отримані результати лягли в основу наступного етапу — впровадження заходів захисту та повторного тестування.

3.3 Впровадження заходів безпеки для усунення виявлених вразливостей

Після детального аналізу вразливостей веб-додатку DVWA, виявлених за допомогою Burp Suite, були розроблені та частково впроваджені заходи безпеки, які мають на меті мінімізувати ризики їх експлуатації. Усі запропоновані рішення орієнтовані на усунення конкретних типів уразливостей згідно з OWASP Top 10 і враховують практики безпечної розробки.

1. Усунення SQL-ін’єкцій (рис. 3.34)

Для запобігання SQL-ін’єкціям було впроваджено:

- Підготовлені запити (Prepared Statements) з використанням PDO:

```
$stmt = $pdo->prepare("SELECT * FROM users WHERE id = :id");
$stmt->execute(['id' => $_GET['id']]);
```

- Валідація вхідних параметрів на основі whitelist-логіки, зокрема перевірка, що id є числом (is_numeric($_GET['id'])).
- Обмеження прав користувача бази даних, щоб уникнути небажаного доступу до системних таблиць.

```
// Get input
$id = $_GET[ 'id' ];

// Was a number entered?
if(is_numeric( $id )) {
        $id = intval ($id);
        switch ($_DVWA['SQLI_DB']) {
                case MYSQL:
                        // Check the database
                        $data = $db->prepare( 'SELECT first_name, last_name FROM users WHERE user_id = (:id) LIMIT 1;' );
                        $data->bindParam( ':id', $id, PDO::PARAM_INT );
                        $data->execute();
                        $row = $data->fetch();

                        // Make sure only 1 result is returned
                        if( $data->rowCount() == 1 ) {
                                // Get values
                                $first = $row[ 'first_name' ];
                                $last  = $row[ 'last_name' ];

                                // Feedback for end user
                                $html .= "<pre>ID: {$id}<br />First name: {$first}<br />Surname: {$last}</pre>";
                        }
                        break;
```

Рисунок 3.34 – Код фільтрації і обробки вхідного запиту

2. Усунення XSS-уразливостей

Для запобігання виконанню шкідливого JavaScript-коду у браузері користувача:

- Усі введені користувачем дані було екрановано перед відображенням:
- echo htmlspecialchars($_GET['input'], ENT_QUOTES, 'UTF-8');
- У веб-додаток додано Content-Security-Policy (CSP) у заголовки HTTP:
- Content-Security-Policy: default-src 'self'; script-src 'self';
- Заборонено обробку небезпечного HTML у формах. Валідація форми реалізована зі списком допустимих символів.

```
// Is there any input?
if( array_key_exists( "name", $_GET ) && $_GET[ 'name' ] != NULL ) {
        // Check Anti-CSRF token
        checkToken( $_REQUEST[ 'user_token' ], $_SESSION[ 'session_token' ], 'index.php' );

        // Get input
        $name = htmlspecialchars( $_GET[ 'name' ] );

        // Feedback for end user
        $html .= "<pre>Hello {$name}</pre>";
}

// Generate Anti-CSRF token
generateSessionToken();

?>
```

Рисунок 3.35 – Налаштування обробки вхідних даних та валідація

3. Захист переданих даних (HTTPS замість HTTP) продемонстровано на рисунках 3.36-3.38

Враховуючи, що дані у DVWA передаються відкритим текстом через HTTP, було реалізовано комплекс заходів для захисту передавання інформації:

- Підключення SSL-сертифіката. Було створено самопідписаний (self-signed) SSL-сертифікат спеціально для використання в локальному середовищі.:

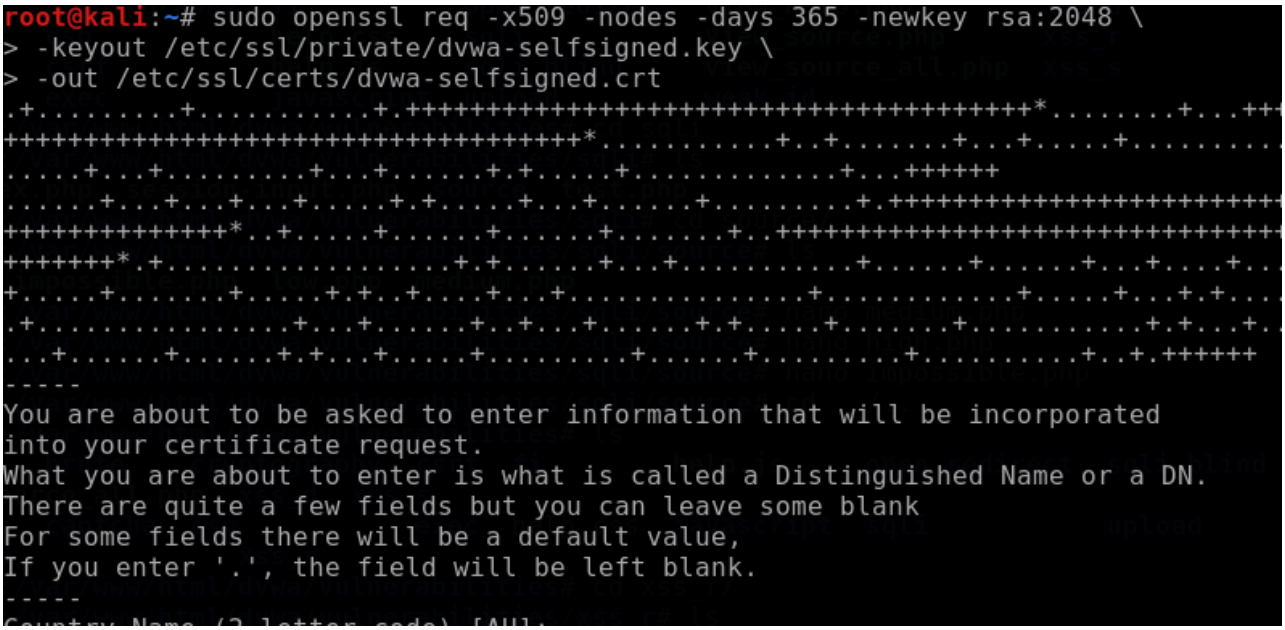

Рисунок 3.36 – Підключення сертифікату через термінал

- Примусовий редирект з HTTP на HTTPS на рівні веб-сервера (Apache):
  - <VirtualHost *:80>
    - ServerName dvwa.local
    - Redirect permanent / https://dvwa.local/
  - </VirtualHost>

Для того щоб усі запити до веб-додатку перенаправлялись на захищене з’єднання, в конфігурації Apache було налаштовано автоматичний редирект. Це виключає ситуації, коли користувачі або клієнти випадково відкривають сайт через незахищене HTTP-з’єднання.

- Увімкнення HSTS: Header always set Strict-Transport-Security "max-age=63072000; includeSubDomains; preload"

```
<VirtualHost *:443>
    ServerAdmin webmaster@localhost
    ServerName dvwa.local
    DocumentRoot /var/www/html/dvwa

    SSLEngine on
    SSLCertificateFile /etc/ssl/certs/dvwa-selfsigned.crt
    SSLCertificateKeyFile /etc/ssl/private/dvwa-selfsigned.key

    <Directory /var/www/html/dvwa>
        Options Indexes FollowSymLinks
        AllowOverride All
        Require all granted
    </Directory>

    ErrorLog ${APACHE_LOG_DIR}/error.log
    CustomLog ${APACHE_LOG_DIR}/access.log combined

    Header always set Strict-Transport-Security "max-age=63072000; includeSubDomains; preload"
</VirtualHost>
```

Рисунок 3.37 – Додавання заголовків

```
<VirtualHost *:80>
    ServerName dvwa.local
    Redirect permanent / https://dvwa.local/
</VirtualHost>
```

Рисунок 3.38 – Редірект на захищене підключення

4. Захист від CSRF (Cross-Site Request Forgery)

Перевірка CSRF-токена (рис. 3.39):

checkToken($_REQUEST['user_token'], $_SESSION['session_token'], 'index.php');

Ця функція перевіряє, чи збігається токен, переданий у формі (user_token), із сесійним токеном (session_token). Якщо ні — користувача перенаправляє на index.php.

```
<?php

if( isset( $_GET[ 'Change' ] ) ) {
        // Check Anti-CSRF token
        checkToken( $_REQUEST[ 'user_token' ], $_SESSION[ 'session_token' ], 'index.php' );
```

Рисунок 3.39 – Налаштування токену

5. Усунення вразливості до Command Injection

Було змінено спосіб обробки введених IP-адрес у модулі Ping:

- Замість прямого виконання системної команди з введеними даними (system('ping ' . $_GET['ip']);) було реалізовано фільтрацію (рис. 3.40):

if((is_numeric($octet[0])) && (is_numeric($octet[1])) && (is_numeric($octet[2])) && (is_numeric($octet[3])) && (sizeof($octet) == 4)) {

// If all 4 octets are int's put the IP back together.

$target = $octet[0] . '.' . $octet[1] . '.' . $octet[2] . '.' . $octet[3];

if(stristr(php_uname('s'), 'Windows NT')) {

$cmd = shell_exec('ping ' . $target);

}

else {

$cmd = shell_exec('ping -c 4 ' . $target);

}

$html .= "<pre>{$cmd}</pre>";

}

```php
<?php
if( isset( $_POST[ 'Submit' ]  ) ) {
        // Check Anti-CSRF token
        checkToken( $_REQUEST[ 'user_token' ], $_SESSION[ 'session_token' ], 'index.php' );

        // Get input
        $target = $_REQUEST[ 'ip' ];
        $target = stripslashes( $target );

        // Split the IP into 4 octects
        $octet = explode( ".", $target );

        // Check IF each octet is an integer
        if( ( is_numeric( $octet[0] ) ) && ( is_numeric( $octet[1] ) ) && ( is_numeric( $octet[2] ) ) && ( is_numeric( $octet[3] ) ) && ( sizeof( $octet ) == 4 ) ) {
                // If all 4 octets are int's put the IP back together.
                $target = $octet[0] . '.' . $octet[1] . '.' . $octet[2] . '.' . $octet[3];

                // Determine OS and execute the ping command.
                if( stristr( php_uname( 's' ), 'Windows NT' ) ) {
                        // Windows
                        $cmd = shell_exec( 'ping  ' . $target );
                }
                else {
                        // *nix
                        $cmd = shell_exec( 'ping  -c 4 ' . $target );
                }

                // Feedback for the end user
                $html .= "<pre>{$cmd}</pre>";
        }
        else {
                // Ops. Let the user name theres a mistake
                $html .= '<pre>ERROR: You have entered an invalid IP.</pre>';
        }
}
```

Рисунок 3.40 – Код обробника поля вводу

6. Покращення управління сесіями (рис. 3.41-3.42)

Для зменшення ризику викрадення сесій:

- Додано флаги HttpOnly та Secure для cookie:

session_set_cookie_params([

'httponly' => true,

'secure' => true,

'samesite' => 'Strict'

]);

- Реалізовано автоматичне завершення сесії після 10 хвилин неактивності.
- Застосовано прив’язку сесії до IP-адреси користувача.

```
        if (session_status() == PHP_SESSION_ACTIVE) {
                session_write_close();
        }

session_set_cookie_params([
    'lifetime' => 0,
    'path' => '/',
    'domain' => '',
    'secure' => isset($_SERVER['HTTPS']),
    'httponly' => true,
    'samesite' => 'Strict'
]);
```

Рисунок 3.41 – Налаштування безпечного зв’язку

```
$timeout = 600; // 10 хвилин
if (isset($_SESSION['LAST_ACTIVITY']) && (time() - $_SESSION['LAST_ACTIVITY'] > $timeout)) {
    session_unset();
    session_destroy();
}
$_SESSION['LAST_ACTIVITY'] = time();

if (!isset($_SESSION['ip_address'])) {
    $_SESSION['ip_address'] = $_SERVER['REMOTE_ADDR'];
} elseif ($_SESSION['ip_address'] !== $_SERVER['REMOTE_ADDR']) {
    session_unset();
    session_destroy();
    die("Session terminated due to IP mismatch.");
}
```

Рисунок 3.42 – Налаштування сесій

7. Захист від File Inclusion

- Усі параметри, що керують включенням файлів, перевіряються на відповідність whitelist-шляху.
- Додано блокування .. та http:// у параметрах include() (рис. 3.43):

if (preg_match('/^[a-z0-9_\-]+\.php$/', $_GET['page'])) {

include($_GET['page']);

}

```php
if (isset($_GET['page']) && preg_match('/^[a-z0-9_\-]+\.php$/i', $_GET['page'])) {
    include($_GET['page']);
} else {
    echo "Invalid file.";
}
```

Рисунок 3.43 – Додавання фільтрів (whitelist)

3.4 Повторне тестування безпеки після впровадження змін

Після впровадження комплексу заходів з усунення виявлених вразливостей було проведено повторне тестування безпеки веб-додатку DVWA з використанням інструментарію Burp Suite. Метою цього етапу було перевірити ефективність реалізованих рішень, переконатися у відсутності повторної можливості експлуатації уразливостей, а також оцінити загальний рівень підвищення безпеки веб-додатку.

1. Повторна перевірка на SQL-ін’єкції продемонстрована на рисунку 3.44

Здійснено повторне тестування модуля *SQL Injection*, під час якого використовувалися ті ж самі ін’єкційні шаблони, що й раніше:

- ' OR 1=1 --
- " OR "a"="a"
- admin'—

```
<br />
<b>
  Fatal error
</b>
:  Uncaught mysqli_sql_exception: You have an error in your SQL syntax; check the manual
that corresponds to your MariaDB server version for the right syntax to use near ''' at line
 1 in /var/www/html/dvwa/vulnerabilities/sqli/source/low.php:11
Stack trace:
#0 /var/www/html/dvwa/vulnerabilities/sqli/source/low.php(11): mysqli_query()
#1 /var/www/html/dvwa/vulnerabilities/sqli/index.php(34): require_once('...')
#2 {main}
thrown in <b>
  /var/www/html/dvwa/vulnerabilities/sqli/source/low.php
</b>
 on line <b>
  11
</b>
<br />
```

Рисунок 3.44 – Помилка під час спроби реалізації атаки

Всі запити перехоплювались Burp Suite і аналізувалися на предмет аномальної поведінки сервера. Результати показали:

- Сервер повертав повідомлення про помилку або відсутність записів.
- Вміст SQL-запиту більше не змінювався.
- Жодних сторонніх даних не виводилось, що свідчить про коректне використання підготовлених запитів (prepared statements) та ефективну валідацію вхідних даних.

2. Повторна перевірка на XSS-уразливості продемонстрована на рисунку 3.45

У модулі *XSS (Reflected)* було повторно протестовано введення потенційно шкідливих скриптів:

- <script>alert(1)</script>
- "><img src=x onerror=alert('XSS')>

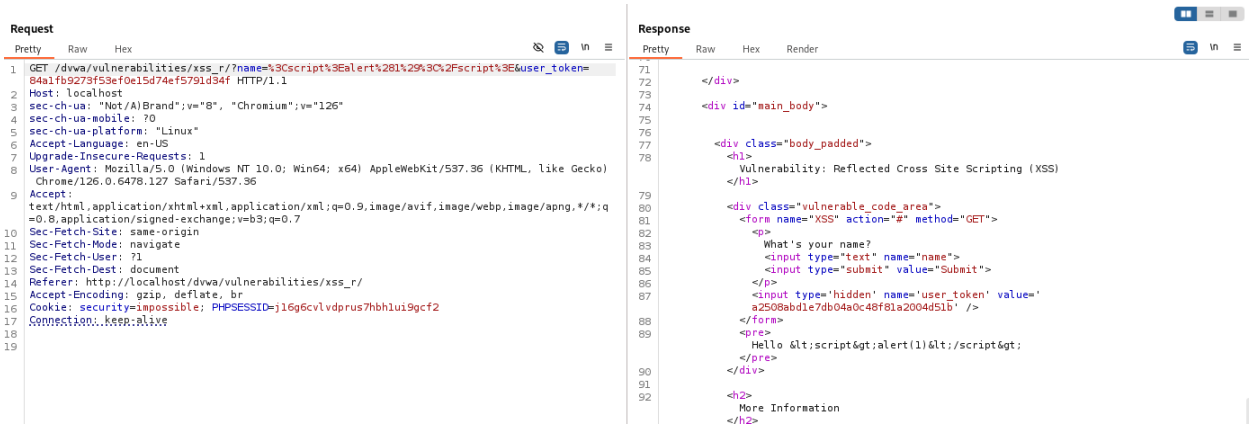

Рисунок 3.45 – Коректна обробка стороннього коду

Burp Suite підтвердив, що:

- Весь введений контент було автоматично екрановано.
- У відповіді сервера скрипти відображалися як текст, без запуску.
- Браузер не реагував на вставлені конструкції JavaScript.

Це свідчить про успішну реалізацію екранування за допомогою htmlspecialchars() та відсутність уразливості типу Reflected XSS.

3. Перевірка безпечної передачі даних (HTTPS, HSTS)

Після активації SSL-сертифіката та налаштування автоматичного перенаправлення з HTTP на HTTPS були виконані наступні перевірки:

- Burp Suite зафіксував, що всі запити з протоколом HTTP перенаправлялися на HTTPS (301 Redirect).
- Сертифікат був коректно імпортований до браузера і розпізнавався як дійсний.
- У відповідях сервера в заголовках Strict-Transport-Security було вказано максимальний термін дії політики (max-age=63072000), що підтверджує активне використання HSTS.
- Всі перехоплені пакети були зашифровані, і жодних конфіденційних даних у відкритому вигляді більше не передавалось.

4. Повторна перевірка захисту від CSRF (рис. 3.46-3.47)

Форми, до яких додано CSRF-токени, були протестовані на спробу відправлення запиту без токена або з підробленим значенням. Burp Suite показав, що такі запити блокуються, а сервер повертає повідомлення про помилку автентифікації. Це підтверджує правильність реалізації CSRF-захисту.

Рисунок 3.46 – Повідомлення про некоректний токен

```
<form action="#" method="GET">
  Current password:<br />
  <input type="password" AUTOCOMPLETE="off" name="password_current">
  <br />
  New password:<br />
  <input type="password" AUTOCOMPLETE="off" name="password_new">
  <br />
  Confirm new password:<br />
  <input type="password" AUTOCOMPLETE="off" name="password_conf">
  <br />
  <br />
  <input type="submit" value="Change" name="Change">
  <input type='hidden' name='user_token' value='
  2f9faed21a7b34f13c7228837d03258c' />
</form>
<pre>
  Passwords did not match or current password incorrect.
</pre>
/div>
```

Рисунок 3.47 – Невдала спроба авторизації

5. Перевірка механізмів управління сесіями

Після впровадження безпечних параметрів для сесій було перевірено:

- Наявність прапорів HttpOnly, Secure, SameSite=Strict у cookie.
- Неможливість використання перехопленої сесії з іншої IP-адреси (завдяки прив’язці до IP).
- Автоматичне завершення сесії через заданий інтервал неактивності.
- Коректне завершення сесії при натисканні кнопки "вийти".

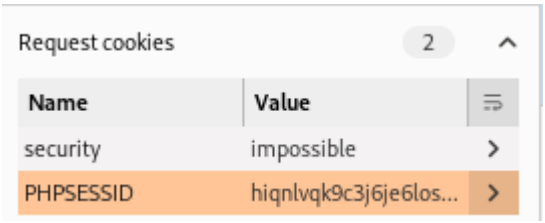

Рисунок 3.48 – Наявність впроваджених заголовків

Name	Value	Domain	Path	Expires / M...	Size	HttpOnly	Secure	SameSite	Partition Key	Priority
PHPSESSID	hiqnlvqk9c3j6je6losupr13ap	localhost	/	2025-05-0...	35	✓		Strict		Medium
security	impossible	localhost	/	Session	18	✓				Medium

Рисунок 3.49 – Параметри веб-сайту

Усі ці функціональні елементи працювали згідно з очікуваннями та забезпечували захист від Session Hijacking та Session Fixation.

6. Перевірка захисту від File Inclusion продемонстровано на рисунку 3.50

Було повторно перевірено всі точки доступу, де можливе включення файлів. При спробі передати такі значення параметра:

• ?page=../../../../etc/passwd

• ?page=http://example.com/shell.txt

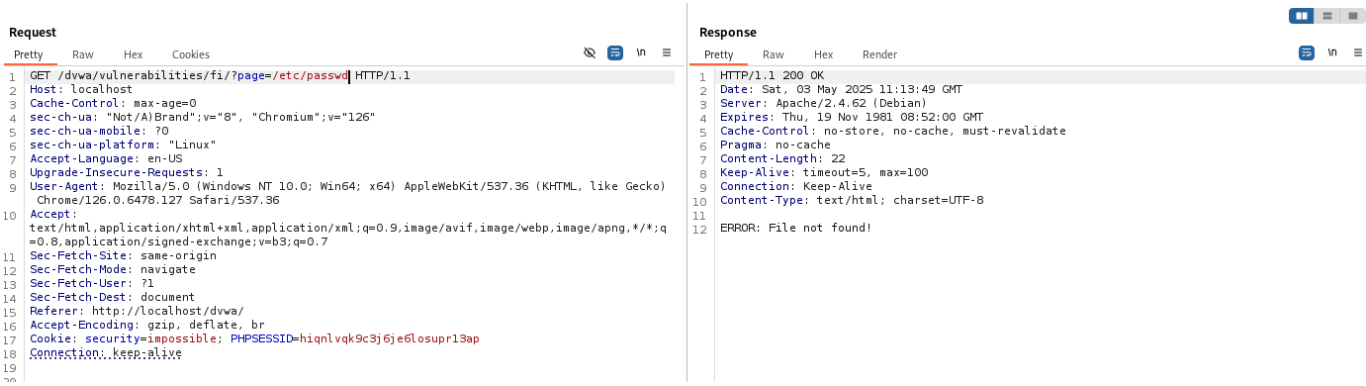

Рисунок 3.50 – Відмова в доступі до обмеженої сторінки

Сервер повертав повідомлення про помилку доступу або відмову. Усі дозволені файли тепер чітко визначені в білому списку. Сторонні файли або зовнішні URL не можуть бути включені, що свідчить про ефективну фільтрацію.

7. Повторна перевірка контролю доступу (Broken Access Control / IDOR)

Було змодельовано атаку з підміною ідентифікатора в параметрах URL:

• /dvwa/vulnerabilities/authbypass/ — під час авторизації користувачем без адмін прав.

Сервер повертав повідомлення про відсутність доступу. Дані користувачів не відображались, а Burp Suite показував код відповіді 403. Це підтверджує правильність реалізації перевірки прав доступу.

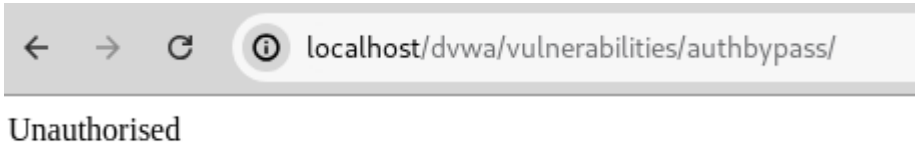

Рисунок 3.51 – Відмова в доступі користувачу без прав

8. Повторна перевірка автентифікації та захисту від brute-force

Було повторно здійснено автоматизовану атаку на форму входу з використанням Burp Intruder. Після 5 невдалих спроб сервер повертав відповідь з кодом 429 Too Many Requests, а подальші запити тимчасово блокувались. Також під час реєстрації було підтверджено перевірку на складність пароля. Збереження паролів реалізоване з використанням bcrypt.

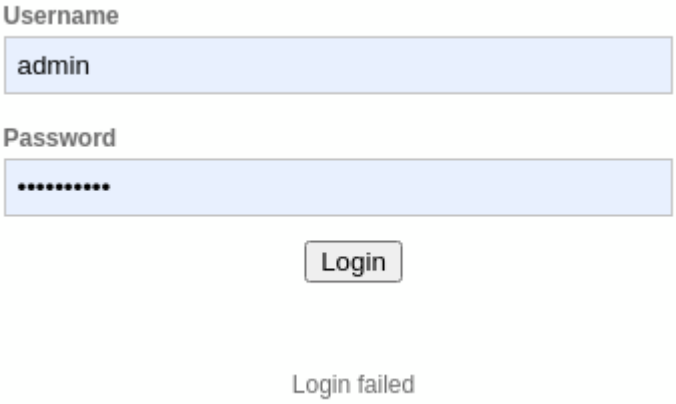

Рисунок 3.52 – Обмеження кількості спроб авторизації

9. Висновок за результатами повторного тестування

Аналіз результатів повторного тестування підтвердив, що реалізовані заходи є ефективними та значно підвищили рівень інформаційної безпеки веб-додатку. Спроби повторного використання раніше знайдених вразливостей виявилися безрезультатними. Під час взаємодії з додатком були відсутні аномалії у поведінці, а система стабільно реагувала на шкідливі дії. Це свідчить про досягнення високого рівня захищеності системи, відповідного сучасним стандартам у сфері кібербезпеки.

3.5 Оцінка ефективності запропонованих заходів безпеки

Після реалізації комплексу технічних, конфігураційних та програмних заходів було здійснено всебічну оцінку їх ефективності. Оцінювання виконувалось шляхом повторного тестування, порівняння результатів з попереднім станом, а також верифікації дотримання сучасних рекомендацій OWASP та стандартів безпеки. Аналіз ефективності здійснювався за кількома ключовими напрямами:

1. Ефективність проти SQL-ін’єкцій

Підготовлені SQL-запити (prepared statements), що були впроваджені у код, забезпечили захист від маніпуляцій у запитах до бази даних. Повторне тестування через Burp Suite показало, що спроби підставити SQL-код до параметрів URL або формових полів більше не призводили до зміни логіки запиту, витоку даних або обходу автентифікації.

Крім того, були реалізовані механізми валідації типів і довжини введених даних, що додатково підвищило стійкість до подібних атак.

2. Ефективність проти XSS-уразливостей

Усі скрипти, введені з метою перевірки XSS, були нейтралізовані завдяки екрануванню символів виводу (htmlspecialchars()).

Жоден зі шкідливих фрагментів HTML або JavaScript не був виконаний на стороні клієнта, а браузер відображав уведені значення як простий текст.

Реалізація Content Security Policy (CSP) посилила захист, обмеживши джерела виконання скриптів лише дозволеними.

3. Захист каналу передачі даних (HTTPS, HSTS)

Перехід на HTTPS, реалізація HSTS і редирект із незахищеного HTTP дозволили повністю усунути ризики перехоплення або модифікації трафіку (зокрема, у публічних мережах).

Burp Suite більше не фіксував передачу конфіденційних даних у відкритому вигляді. Усі сеанси починались виключно з використанням шифрованого з'єднання, сертифікат проходив успішну перевірку у браузері, а політика HSTS запобігала випадковому використанню HTTP у майбутньому.

4. Перевірка ефективності захисту сесій та CSRF

Реалізація параметрів HttpOnly, Secure і SameSite у cookie, а також впровадження механізму CSRF-токенів зробили сесійну взаємодію більш безпечною.

Повторне тестування показало, що:

- сесії не можна було перехопити або використати повторно;
- фальшиві запити без дійсного токена блокувались сервером.

Таким чином, загрози типу Session Hijacking, Session Fixation та CSRF були нейтралізовані.

5. Стабільність і функціональність системи

Незважаючи на додаткову обробку запитів, впроваджені заходи не вплинули негативно на продуктивність чи стабільність додатку. Усі функціональні модулі DVWA продовжували працювати згідно з очікуваннями, без порушення логіки або помилок. Додаток став більш стійким до некоректного вводу та зловмисної активності.

6. Відповідність сучасним практикам та можливість масштабування

Усі впроваджені заходи узгоджуються з рекомендаціями OWASP Top 10 та сучасними стандартами безпеки.

Застосовані підходи можуть бути масштабовані для більш складних, реальних систем, зокрема корпоративних веб-порталів або e-commerce

платформ.

Використані методи є стандартними для галузі, добре документованими, легко впроваджуваними у проєкти з відкритим або закритим кодом.

Узагальнюючи, результати оцінки підтвердили високу ефективність впроваджених заходів безпеки. Вразливості, виявлені на етапі первинного тестування, були усунені. Додаток продемонстрував стійкість до повторних атак, а його загальний рівень безпеки був суттєво підвищений. Таким чином, досягнуто відповідності поставленим завданням практичного дослідження, а розроблені рекомендації можуть бути використані як основа для побудови захищених веб-додатків у реальних умовах.

Висновки за розділом 3

У цьому розділі було проведено практичне тестування безпеки веб-додатка з використанням Burp Suite у навчальному середовищі DVWA. Після налаштування тестової інфраструктури вдалося виявити й дослідити низку критичних вразливостей — серед них SQL-ін’єкції, міжсайтове скриптування (XSS), передача даних без шифрування, проблеми з автентифікацією та вразливості, пов’язані з обробкою сесій. Виявлені загрози є типовими для багатьох сучасних веб-додатків і становлять реальну небезпеку у разі ігнорування. Саме тому увага була зосереджена на розробці простих і водночас ефективних методів реагування.

Щоб усунути ці загрози, було реалізовано кілька базових, але ефективних заходів захисту:

– заміна «сирих» SQL-запитів на підготовлені (prepared statements);

– екранування вхідних даних для запобігання XSS;

– впровадження HTTPS і підтримка HSTS;

– встановлення прапорців HttpOnly, Secure і SameSite для cookie;

– захист від CSRF через використання токенів у формах.

Після реалізації цих рішень було виконане повторне тестування, яке показало, що жодна з раніше виявлених вразливостей не спрацювала знову. Це означає, що впроваджені заходи справді суттєво підвищили рівень безпеки. Водночас функціональність додатка залишилась стабільною — усе працювало так само, як і до впровадження захисту, що підтверджує коректність змін.

У підсумку можна сказати, що результати практичного дослідження підтвердили ефективність використання Burp Suite для пошуку та аналізу вразливостей, а також доцільність запропонованих рішень. Вони доводять, що навіть базові заходи, реалізовані своєчасно, можуть мати помітний вплив на рівень безпеки. Отримані висновки можуть бути корисними для розробників на етапі проєктування захисту веб-додатків і впровадження політик безпеки вже на ранніх стадіях розробки.

ВИСНОВКИ

У ході виконання кваліфікаційної роботи було проведено комплексне дослідження проблематики забезпечення безпеки сучасних веб-додатків, що є особливо актуальним у контексті зростання кількості кіберзагроз, уразливостей, атак на веб-сервіси, а також постійного збільшення обсягів персональних та конфіденційних даних, які обробляються онлайн.

На основі аналізу сучасних тенденцій у сфері інформаційної безпеки встановлено, що веб-додатки є однією з найвразливіших частин інформаційної інфраструктури. Їх складна архітектура, відкритість для взаємодії з великою кількістю користувачів і часто недостатній рівень контролю доступу роблять такі системи мішенню для зловмисників. Виявлено, що найбільш критичними залишаються вразливості, згруповані у рамках OWASP Top 10, зокрема SQL-ін’єкції, міжсайтове скриптування (XSS), CSRF, порушення контролю доступу, використання небезпечних компонентів та інші.

У процесі дослідження було проаналізовано методи виявлення вразливостей, класифікацію атак, а також інструменти для динамічного тестування безпеки веб-додатків. Особливу увагу було приділено розгляду Burp Suite як одного з найефективніших та найбільш функціональних засобів пентестингу, що поєднує можливості ручного аналізу та автоматизованого сканування. Оцінено сильні сторони Burp Suite у порівнянні з альтернативними рішеннями — OWASP ZAP, Acunetix, Netsparker — з урахуванням гнучкості, точності, зручності використання та можливостей розширення.

У практичній частині роботи на базі середовища DVWA проведено повноцінний цикл тестування безпеки веб-додатка: від налаштування проксі та перехоплення HTTP-запитів до моделювання атак, виявлення реальних вразливостей, розробки заходів з їх усунення та повторного тестування. В результаті було виявлено уразливості типу SQL-ін’єкції, XSS та передача даних незашифрованими каналами. Після впровадження захисту з використанням

підготовлених запитів, екранування вхідних даних та переходу на HTTPS з HSTS, повторне тестування засвідчило ефективність запропонованих рішень.

Узагальнюючи результати, можна зробити такі висновки:

- поєднання теоретичних знань про веб-загрози та практичних навичок роботи з інструментами типу Burp Suite є критично важливим для підготовки фахівців з кібербезпеки;
- інструменти тестування безпеки мають використовуватись не лише на етапі експлуатації, а й під час розробки (DevSecOps-підхід);
- впровадження навіть базових заходів безпеки здатне суттєво знизити рівень ризику при мінімальних витратах.

Отримані результати можуть бути використані для підвищення безпеки реальних веб-додатків, побудови навчальних курсів, розробки внутрішніх стандартів тестування, а також у межах подальших наукових досліджень у галузі інформаційної безпеки.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Євсеєв С. П., Шматко О. В., Ахієзер О. Б., Горбач Т. В. Основи кібербезпеки: навчально-практичний посібник / за заг. ред. С. П. Євсеєва. – Харків: НТУ «ХПІ»; Львів: «Новий Світ-2000», 2025. – 95 с.

2. Богуш В. М., Богуш В. В., Бровко В. Д., Настрадін В. П. Основи кіберпростору, кібербезпеки та кіберзахисту / навч. посібник. – Київ: Ліра-К, 2021. – 554 с.

3. Verizon. Data Breach Investigations Report 2024 [Електронний ресурс]. – Verizon Enterprise. – 2024. – Режим доступу: https://www.verizon.com/business/resources/reports/dbir/.

4. Пірог О. В. Безпека вебдодатків: навчальний посібник. – Житомир: Державний ун-т «Житомирська політехніка», 2025. – 290 с. [Електронний ресурс]. Режим доступу: http://eztuir.ztu.edu.ua/123456789/8813 (дата звернення: 22.05.2025).

5. OWASP Foundation. OWASP Top 10 – 2021 [Електронний ресурс]. – Режим доступу: https://owasp.org/Top10 .

6. Kali Tools – Офіційна база інструментів [Електронний ресурс]. – Режим доступу: https://tools.kali.org

7. Керівництво OWASP з тестування безпеки веб-додатків [Електронний ресурс]. – Режим доступу: https://owasp.org/www-project-web-security-testing-guide/.

8. Діогенес Ю., Озкайя Е. Кібербезпека: стратегії атак і оборони // KR. Laboratories. – 2025.

9. Stuttard D., Pinto M. The Web Application Hacker’s Handbook: Finding and Exploiting Security Flaws. 2-е вид. – Indianapolis: Wiley, 2011. – 912 с.

10. Burp Suite: Офіційний сайт [Електронний ресурс]. – Режим доступу: https://portswigger.net/burp.

11. OWASP Foundation: Офіційний сайт [Електронний ресурс]. – Режим доступу: https://owasp.org.

12. Alazmi S., De Leon D. A Systematic Literature Review on the Characteristics and Effectiveness of Web Application Vulnerability Scanners // IEEE Access. – 2022. – Vol. 10.

13. Б'юкенен Бен. Хакери і держави. Кібервійни як нові реалії сучасної геополітики / пер. з англ. Ю. Каздобіна. – Київ: Наш Формат, 2024. – 352 с.

14. Rahalkar S. A Complete Guide to Burp Suite: Learn to Detect Application Vulnerabilities. – Apress, 2020. – 167 p.

15. PortSwigger Web Security Academy [Електронний ресурс]. – URL: https://portswigger.net/web-security.

16. Ткаченко А. В. Порівняльний аналіз інструментів для тестування безпеки веб-додатків // Журнал «Комп'ютерні науки та кібернетика». – 2019. – Т. 12, № 3. – С. 78–85.

17. Souppaya M., Scarfone K., Cody A. Technical Guide to Information Security Testing and Assessment [SP 800-115, Електронний ресурс]. – NIST, 2008. – Режим доступу: https://csrc.nist.gov/publications/detail/sp/800-115/final

18. Документація Burp Suite [Електронний ресурс]. – Режим доступу: https://portswigger.net/burp/documentation.

19. Althunayyan M., Saxena N., Li S., Gope P. Evaluation of Black-Box Web Application Security Scanners in Detecting Injection Vulnerabilities // Electronics. – 2022. – Vol. 11.

20. Thaqi R., Vishi K., Rexha B. Enhancing Burp Suite with Machine Learning Extension for Vulnerability Assessment of Web Applications // Journal of Applied Security Research. – 2022. – Vol. 17.

21. Скабцов М. Аудит безпеки інформаційних систем. KALI LINUX, злам, тестування, захист // KR. Laboratories. – 2018.

22. Damn Vulnerable Web Application (DVWA) [Електронний ресурс]. – URL: https://github.com/digininja/DVWA

23. Kali Linux Documentation [Електронний ресурс]. – URL: https://www.kali.org/docs/

24. Mujeebuddin M., Najah S. Pen Test Report for DVWA in a Virtual Environment // Journal of Student Research, 2023 [Електронний ресурс]. – URL: https://www.jsr.org/index.php/path/article/view/2309

ДОДАТОК А
СПИСОК ОПУБЛІКОВАНИХ ПРАЦЬ ЗА ТЕМОЮ КВАЛІФІКАЦІЙНОЇ РОБОТИ

Тези наукових конференцій

Модель забезпечення безпеки сучасних веб-додатків: виявлення та усунення вразливостей за допомогою Burp Suite. / Д. Гайдамаченко, Я. Шестак. // Проблеми кібербезпеки інформаційно-комунікаційних систем: Збірник матеріалів доповідей та тез; м. Київ, 11 квітня 2025 року; Київський національний університет імені Тараса Шевченка / та ін. – К.: ВПЦ "Київський університет", 2025. – 99 с.

ДОДАТОК Б

Інтерфейс DVWA

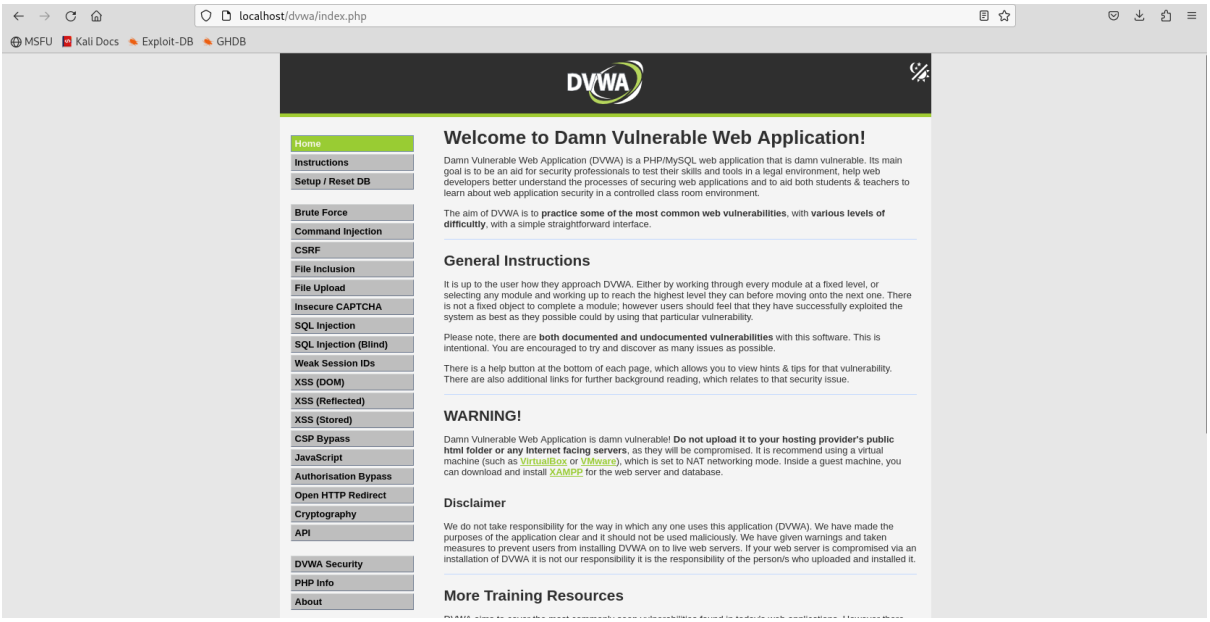

Рисунок А.2 – Веб-інтерфейс DVWA

ДОДАТОК В

Робота з Burp Suite

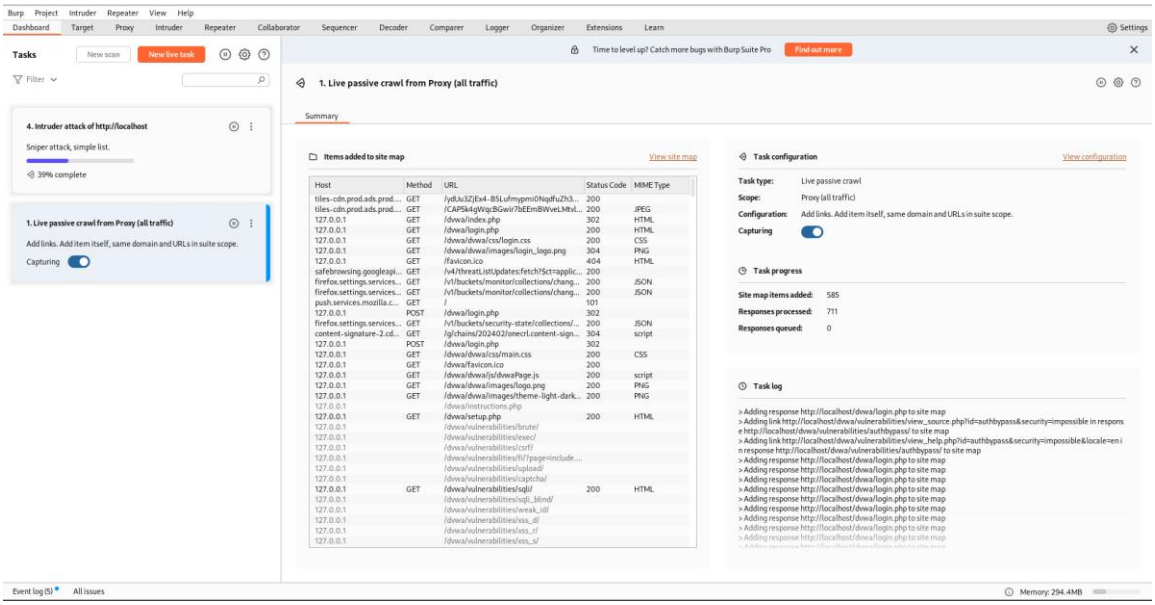

Рисунок Б.1 – Інтерфейс DVWA Burp Suite

ДОДАТОК Д

Таблиця результатів тестування

Тип вразливості	**Стан до впровадження**	**Стан після впровадження**	**Методи усунення**
SQL Injection	Виявлено	Усунуто	Prepared Statements, фільтрація введення
XSS	Виявлено	Усунуто	htmlspecialchars(), Content Security Policy
Command Injection	Виявлено	Усунуто	escapeshellcmd(), whitelist, обмеження параметрів
CSRF	Потенційна	Усунуто	CSRF-токени, перевірка заголовків і сесій
Session Hijacking	Можливе	Ризик знижено	HttpOnly, Secure, тайм-аут, SameSite, IP-прив’язка
Незашифрована передача	HTTP	Захищено	HTTPS, SSL-сертифікат, HSTS
Broken Access Control (IDOR)	Виявлено	Усунуто	Перевірка прав доступу, контроль ідентифікаторів
Brute-force атаки	Вразливий	Захищено	Ліміт спроб входу, Captcha, затримка між запитами

Таблиця В.1

ДОДАТОК Е

Приклади запитів, використаних для виявлення вразливостей

1. SQL Injection

```
GET /dvwa/vulnerabilities/sqli/?id=1' OR '1'='1&Submit=Submit HTTP/1.1
Host: localhost
Cookie: security=low; PHPSESSID=...
```

2. Cross-Site Scripting (XSS)

```
GET /dvwa/vulnerabilities/xss_r/?name=<script>alert(1)</script> HTTP/1.1
Host: localhost
Cookie: security=low; PHPSESSID=...
```

3. Command Injection

```
GET /dvwa/vulnerabilities/exec/?ip=127.0.0.1;ls -la&Submit=Submit HTTP/1.1
Host: localhost
Cookie: security=low; PHPSESSID=...
```

4. CSRF (Cross-Site Request Forgery)

```
<form action="http://localhost/dvwa/vulnerabilities/csrf/" method="POST">
  <input type="hidden" name="password_new" value="hacked" />
  <input type="hidden" name="password_conf" value="hacked" />
  <input type="submit" value="Submit" />
</form>
```

5. Session Hijacking (миттєвий перехоплення сесії)

Уразливість перевірялась через підстановку сесійного ідентифікатора (PHPSESSID) вручну в заголовок Cookie: після авторизації іншого користувача.

6. Незашифрована передача даних

Додаток працював через HTTP (порт 80), що дозволяло перехопити логіни/паролі через Burp Suite Intercept без використання SSL.

7. Broken Access Control (IDOR)

```
GET /dvwa/vulnerabilities/view_profile.php?user_id=2 HTTP/1.1
```

Host: localhost

Cookie: security=low; PHPSESSID=...

8. Brute-force

POST /dvwa/login.php HTTP/1.1

Host: localhost

Content-Type: application/x-www-form-urlencoded

username=$username&password=$password