НАЦІОНАЛЬНИЙ УНІВЕРСТИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ

Факультет Інформаційних технологій

ПОГОДЖЕНО
Декан факультету Інформаційних технологій

_____ Ігор БОЛБОТ
(підпис)

“___” _____ 2025 р.

ДОПУСКАЄТЬСЯ ДО ЗАХИСТУ
Завідувач кафедри комп’ютерних систем, мереж та кібербезпеки

_____ Дмитро КАСАТКІН
(підпис)

“___” _____ 2025 р.

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему «Дослідження атаки на комп’ютерну систему з використанням SQL-ін’єкції»

Спеціальність 123 «Комп’ютерна інженерія»

Освітня програма «Комп’ютерні системи захисту інформації»

Орієнтація освітньої програми освітньо-професійна

Гарант освітньої програми

д.п.н., професор _____ Сергій МАМЧЕНКО
(підпис)

Керівник магістерської кваліфікаційної роботи

д.т.н., професор _____ Валерій ЛАХНО
(підпис)

Виконав _____ Андрій ЯСІЛЬОНІС
(підпис)

КИЇВ – 2025

НАЦІОНАЛЬНИЙ УНІВЕРСТИТЕТ БІОРЕСУРСІВ
І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ

Факультет Інформаційних технологій

ЗАТВЕРДЖУЮ
Завідувач кафедри комп'ютерних систем, мереж та кібербезпеки
К.п.н., доцент _____ Дмитро КАСАТКІН
(підпис)

«____» _____2025 року

З А В Д А Н Н Я

ДО ВИКОНАННЯ МАГІСТЕРСЬКОЇ КВАЛІФІКАЦІЙНОЇ РОБОТИ СТУДЕНТУ

Ясільоніс Андрій Вінцасович

Спеціальність **123** «Комп'ютерна інженерія»
Освітня програма Комп'ютерні системи захисту інформації
Орієнтація освітньої програми освітньо-професійна
Тема магістерської кваліфікаційної роботи «Дослідження атаки на комп'ютерну систему з використанням SQL-ін'єкції»
затверджена наказом ректора НУБіП України від «29» жовтня 2024 р. №1941 «С»
Термін подання завершеної роботи на кафедру _____
(рік, місяць, число)
Вихідні дані до магістерської кваліфікаційної роботи

1. Результати експлуатації веб-вразливості SQL-ін'єкції в лабораторних умовах
2. Впровадження та аналіз систем моніторингу і засобів захисту середовища
3. Розрахунки та дослідження метриків ефективності захисту веб-застосунку в умовах атаки

Перелік питань, що підлягають дослідженню:

1. Аналіз предметної області
2. Розробка не захищеного веб-середовища для тестування
3. Впровадження системи моніторингу журналів подій та засобів захисту веб-застосунку
4. Експлуатація вразливості застосунку до та після впровадження захисту
5. Аналіз отриманих результатів

Дата видачі завдання «_____»_____2024 р.

Керівник магістерської кваліфікаційної роботи _____ Валерій ЛАХНО
(підпис)

Завдання прийняв до виконання _____ Андрій ЯСІЛЬОНІС
(підпис)

РЕФЕРАТ

Актуальність магістерської роботи зумовлена стрімким розвитком веб-технологій та цифровізацією бізнес-процесів організацій, що активному створенню веб-застосунків, які безспосередньо взаємодіють із базами даних, де зберігається чутлива інформація. Саме отримання доступу до таких даних є основною метою зловмисників, а однією з найпоширеніших вразливостей, завдяки якій можна отримати несанкціонований доступ до бази даних та її даних, є SQL-ін’єкція. У межах даної роботи досліджено принципи здійснення подібних атак, засоби моніторингу та захисту веб-застосунків, методи та інструменти, якими зловмисник може скористатися.

Об’єктом дослідження є розроблений веб-застосунок, інтегрований з базою даних PostgreSQL.

Предметом дослідження є тестування вразливості до SQL-ін’єкцій, механізми виявлення та захисту.

Метою дослідження є аналіз засобів реалізації атак типу SQL-ін’єкція, а також оцінка ефективності засобів моніторингу та захисту в межах контрольованого лабораторного середовища.

Для досягнення мети дослідження визначено перелік завдань:

- аналітичний огляд SQL-ін’єкцій;
- аналіз засобів виявлення та захисту SQL-ін’єкцій;
- розробити середовище з базою даних SQL;
- провести імітацію атаки в контрольованих умовах;
- впровадити засоби захисту та моніторингу веб-ресурсів;
- оцінити використанні засоби захисту та моніторингу в умовах експлуатації вразливості.

Практична цінність кваліфікаційної роботи полягає у підвищенні рівня захищенності інформації від однієї з найпоширеніших видів атак на веб-ресурси.

Своєчасне виявлення та блокування шкідливих запитів може запобігти витоку конфіденційної інформації, яка зберігається в організації, або її втраті, що напряму впливає на бізнес-процеси компанії, як ключовий аспект кібербезпеки.

Магістерська робота складається зі вступу, трьох розділів, висновків, списку використаних джерл з 31 найменуванням та трьох додатків. Загальний обсяг становить 65 сторінок, з яких 55 сторінок основного тексту. Робота містить ілюстрований матеріал, а саме 23 рисунки та 5 таблиць.

Ключові слова:

SQL-ін’єкція, веб-застосунок, вразливість, моніторинг, безпека.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

КС	– Комп’ютерна система
SQL	– Structured Query Language
SQLi	– SQL-ін’єкція
TCP	– Transmission Control Protocol
PHP	– Hypertext Preprocessor
WAF	– Web Application Firewall
Intercept Proxy	– Перехоплюючий проксі для зміни HTTP-трафіку
CVE	– Common Vulnerabilities and Exposures
DAST	– Dynamic Application Security Testing
IAST	– Interactive Application Security Testing
RCE	– Remote Code Execution
RFE	– Remote File Execution
TP	– True Positive
TN	– True Negative
FP	– False Positive
FN	– False Negative

ВСТУП

Сучасний світ розвивається стрімко, а технологічний прогрес охоплює практично всі сфери діяльності людини. Висока ефективність роботи, швидкість обміну інформацією та доступність ресурсів стали важливими складовими нашого життя. Універсальним інструментом для зберігання, швидкої передачі та обробки даних став Інтернет, який забезпечує комунікацію між мільйонами користувачів у всьому світі.

Стрімке зростання обсягів інформації та кількості онлайн-сервісів супроводжується зростанням кіберзагроз, де Інтернет стає основним середовищем для більшості відомих кібератак, спрямованих на отримання несанкціонованого доступу, пошкодження або виведення з ладу комп'ютерних систем і мереж. Вразливості у веб-додатках, які забезпечують роботу більшості сучасних сервісів, стають потенційними точками входу для зловмисників. Це створює небезпеку для трьох ключових принципів кібербезпеки — **конфіденційності**, **цілісності** та **доступності** інформації.

Порушення цих принципів може нести серйозні наслідки. Витік або знищення даних призводить до фінансових збитків, втрати репутації організацій, і навіть може становити загрозу національній безпеці. Якщо організація не здатна забезпечити належний захист своїх інформаційних активів, це підриває довіру клієнтів, а також ставить під сумнів її надійність.

Більшість відомих вразливостей, які використовуються у атаках були виявлені у коді веб-додатків. І найкращий захист – створення безпечних додатків. Існує безліч проєктів спрямованих на вивчення веб-атак та навчання розробників уникати відомих вразливостей коду, зокрема відомий у всьому світі проєкт Open Web Application Security Project (OWASP) [1]. Організація кожні три роки публікує список десяти самих популярних веб-атак. Остання публікація 2021 року ставить

атаку Injection на третє місце, а в 2017 році вона займала перше місце. Самий такий вид атаки може поєднювати вразливість як веб-серверу чи додатку, так і базу даних. Однією з самих відомих атак типу «ін'єкція» є SQL-ін'єкція, яка досі є актуальною загрозою для веб-додатків.

Об'єктом дослідження є веб-застосунок із базою даних PostgreSQL.

Предметом дослідження є тестування вразливості SQL-ін'єкції, механізми виявлення та захисту від SQL-ін'єкцій.

Метою дослідження є аналіз засобів реалізації SQL-ін'єкцій, а також оцінка ефективності заходів моніторингу та захисту на прикладі атаки на контрольоване локальне середовище.

Для досягнення мети дослідження встановлено перелік завдань:

- аналітичний огляд SQL-ін'єкцій;
- аналіз засобів виявлення та захисту SQL-ін'єкцій;
- розробити середовище з базою даних SQL;
- провести імітацію атаки в контрольованих умовах;
- впровадити засоби захисту та моніторингу веб-ресурсів;
- оцінити використанні засоби захисту та моніторингу в умовах експлуатації вразливості.

Практична цінність цієї кваліфікаційної роботи полягає в підвищенні рівня захисту інформації від однієї з найпоширеніших видів атак на веб-ресурси. Моніторинг і блокування шкідливих запитів може запобігти витоку конфіденційної інформації, яка зберігається в організації, або її втраті, що напряму впливає на бізнес-процеси компанії, як ключовий аспект кібербезпеки.

РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Визначення і класифікація SQL-ін’єкцій

Веб-додатки, сайти та онлайн-платформи, що створенні для забезпечення зручності і доступності для користувачів, формують широку поверхню потенційних атак для зловмисників. Веб-атака – це тип кіберзагрози, коли зловмисники використовують існуючу вразливість веб-сервісу з метою порушення його роботи, викрадення конфіденційних даних або поширення шкідливого програмного забезпечення. Найбільш поширеними атаки є на веб-додатки, що включають у собі веб-сервер, клієнтську частину додатку (веб-сторінки) та базу даних для збереження інформації.

Атака з використанням SQL-ін’єкції цілеспрямована на вразливість коду веб-додатку, що дозволяє отримати несанкціонований доступ до бази даних.

Ін’єкція SQL – це вразливість веб-безпеки, яка дозволяє зловмисникові втручатися в запити, які додаток надсилає до своєї бази даних. Це дає можливість атакуючому виконати довільний запит до бази даних (наприклад, прочитати вміст будь-яких таблиць, видалити, змінити або додати дані до таблиці), отримати можливість читання та/або запису локальних файлів та виконання довільних команд на сервері [2].

Розглянемо приклад SQL-ін’єкції, що експлуатує вразливий запит до бази даних:

*SELECT * FROM users WHERE username = ‘$_POST[‘username’]’ AND password = ‘$_POST[‘password’]’;*

Використовуючи поле для введення (*‘username’*), зловмисник може модифікувати запит, додавши шкідливий код:

’OR 1=1; --

Де до бази даних відправиться наступний запит:

*SELECT * FROM users WHERE username = '' OR 1=1; --' AND password = '$_POST['username']';*

Проаналізувавши цей запит, можна побачити, що запит завершує свою обробку на логічній умові "1=1", так як в синтаксисі SQL "--" використовується для початку коментаря, і наступні символи не враховуються. Отже, виконання цього запиту "обманює" застосунок, та повертає всі записи таблиці, незалежно від того, які облікові дані були введені користувачем.

Наслідки SQL-ін'єкції можуть включати:

- витік чутливої та конфіденційної інформації;
- зміна та/або видалення інформації;
- несанкціонований доступ до системи;
- репутаційні ризики

Існує 3 основні категорії SQL-ін'єкцій:

- in-band SQL-ін'єкція;
- inferential SQL-ін'єкція;
- out-of-band SQL-ін'єкція.

Проаналізувавши основні категорії і види SQL-ін'єкцій, можна зрозуміти, чому вони є доволі поширеними, та небезпечними. Нижче наведено основну класифікацію атак.

In-band SQLi (або класичні SQL-ін'єкції)

Даний вид ін'єкцій є найпоширенішим та простим для експлуатації. In-band ін'єкція означає коли зловмисник використовує, або може використовувати, один і

той самий канал як для атаки, так і для ексфільтрації [3]. Найбільш поширені типи in-band SQL-ін’єкцій зазначено в таблиці 1.1:

Таблиця 1.1

Типи In-band SQL-ін’єкцій

Тип SQL-ін’єкції	Опис
Error-based SQLi	Метод ін’єкції, який розрахований на повідомлення про помилки, які відправляє сервер бази даних для отримання інформації про структуру базу даних.
Union-based SQLi	Техніка ін’єкції, яка використовуючи оператор мови SQL “UNION” може об’єднати в один результат декілька операторів “SELECT” одночасно, який потім повертається як частина HTTP-відповіді.

Inferential SQLi (blind SQL, або сліпа SQL-ін’єкція)

Категорія SQL-ін’єкцій, що поєднує два типи “сліпих” SQL-ін’єкцій. Цей тип ін’єкцій є найбільш повільним, так як зловмиснику потрібно перебрати всю базу даних вручну[3]. Два різновиди цієї категорії вказано в таблиці 1.2:

Таблиця 1.2

Типи “сліпих” SQL-ін’єкцій

Тип SQL-ін’єкції	Опис
Boolean-based SQLi	Метод ін’єкції, який відправляє SQL-запит до бази даних, з логічною умовою, та очікує на повернення результатів TRUE або FALSE типу.

Time-based SQLi	Техніка ін’єкції, де в тілі SQL-запиту вбудовано код, що заставляє базу даних чекати певний проміжок часу, перед тим як відповісти на нього. Час очікування вказує на факт обробки такого запиту, тобто дає зловмиснику зрозуміти, яким чином обробляються запити на сервері.

Out-of-band SQLi

Остання категорія SQL-ін’єкцій менш популярна за свої аналоги, та експлуатується у виключних випадках, коли зловмисник не може використовувати один і той же канал для атаки і для ексфільтрації, а також залежить від функцій, включених на сервері.

Ця категорія атаки не має конкретних типів, так як використовується у окремих випадках, які залежать від серверу бази даних. Out-of-band SQL-ін’єкція зазвичай розрахована на можливість робити DNS або HTTP-запити, для відправки значень на підконтрольний зловмиснику сервер [3].

1.2 Механізми виникнення вразливості

Атака з використанням SQL-ін’єкції виконується при умові наявної вразливості у скриптах серверу, де зловмисник вивчає їх поведінку за допомогою маніпуляції вхідними даними, та спостерігає за аномальною поведінкою. Маніпуляція може здійснюватися різними параметрами:

- з використанням POST та GET запитів;
- значенням cookie в HTTP-запитах;
- HTTP заголовками REFERER, що використовуються для переадресації на іншу сторінку;
- експлуатацією методів автентифікації.

Окрім розвідки вразливості у обробці запитів скриптами серверу, існують також інші причини виникнення такої вразливості, які виникають через недотримання безпеки коду розробниками та адміністраторами [4]:

- відсутність валідації вхідних даних, яких формують SQL-запит;
- не обмежений доступ до бази даних, зокрема користувачів з правами адміністраторів або правами-root;
- відсутність або недостатній рівень захисту системи автентифікації;
- відсутність або нехтування заходами по моделюванню загроз, використання неоновлених/застарілих технологій та систем;
- свідоме або не свідоме використання готових рішень з низьким рівнем захисту від атак;

Розберемо на прикладі атаку, яка використовує вразливість SQL-запитів. У випадку, якщо розробником або організацією використовується комерційне програмне забезпечення, або з відкритим кодом (для прикладу, розміщене на Github), зловмисник може визначити версію системи або додатку, проаналізувати механізми SQL-запиту та HTTP-відповіді, і експлуатувати дану вразливість.

Уявімо, зловмисник намагається отримати доступ до інформації бази даних інтернет-магазину, що використовує типову архітектуру додатку веб-клєінт → веб-сервер → база даних SQL, де повертає дані у вигляді HTTP-відповіді. Інтернет магазин має декілька таблиць, зокрема таблиці товарів, та таблиці зареєстрованих користувачів магазину. Використовуючи цю інформацію зловмисник може провести UNION атаку:

В каталозі товарів додаток виконує запит за пошуком необхідної категорії:

SELECT name, price, description FROM products WHERE category = ‘Phone’

Зловмисник може у полі пошуку вписати корисне навантаження, та отримати доступ до інших таблиць бази даних:

’ UNION SELECT username, password FROM users --

Сервер опрацює перший запит, який повертає список товарів з категорії “Phone”, а далі опрацьовує оператора UNION, який може об’єднувати декілька запитів одночасно, та виводить у результаті всі паролі та логіни користувачів магазину.

Цілісність системи не була пошкоджена, додаток не зазнав ніяких змін у своїй роботі, проте ця вразливість дозволила вивести окрім легітимної та публічної інформації – конфіденційну, з обмеженим доступом.

1.3 Методи виявлення та захисту від SQL-ін’єкцій

Виявленння вразливостей може здійснюватися в ручному режимі або автоматизованим способом. Перевірка вручну вимагає необхідних знань та розуміння як вразливості, так і принцип роботи системи, взаємодія додатку з сервером. Зазвичай для таких дій наймають спеціалізованого фахівця, сфера діяльності якого – пентест. Проте, є деякі широковідомі способи ручного пошуку вразливості коду або додатку, на які варто звернути увагу [5]:

1. Тестування різних сценаріїв, зокрема введення спецсимволів, вивчення та аналіз реакції додатку, які виникають при цьому помилки, чи є зворотні відповіді, що могли б дати зловмиснику корисну інформацію про систему.
2. Перевірка виведення результатів обробки запиту, чи відображаються вони на веб-сторінці, або в повідомленнях про помилку.
3. Відправка запитів, які створюють часові затримки при виконанні, пошук відмінностей в часі. Це дозволить зловмиснику зрозуміти, які запити база даних успішно обробляє.

4. Тестування взаємодії з формами та інтерактивними елементами веб-сайту, куди є можливість вводити дані, зокрема форми автентифікації, пошуку та інше.
5. Аналіз обробки неправильних вхідних даних, та як веб-сервер на них реагує. Якщо він не екранує та не відкидає їх, значить це створює точку для потенційної атаки зловмисника.

Автоматизований пошук вразливостей

Сканери вразливостей є найбільш ефективним та дієвим способом пошуку вразливостей. Такі інструменти використовуються для багатьох типів систем, проте ми розглянемо декілька сканерів пошуку вразливостей SQL-ін’єкції у веб-додатках [6].

SQLMap – один із самих популярних інструментів, який автоматизовано впроваджує SQL-ін’єкції. SQLMap використовується для тестування серверів баз даних, може витягувати імена бази даних, таблиці, стовпці, а також читати та редагувати файли на віддаленій файловій системі. Такі характеристики роблять його потужним інструментом, для етичного хакерства і пошуку вразливостей для їх усунення.

Burp Suite – програмний комплекс набору інструментів та призначений для аудиту безпеки веб-додатків. Один із самих функціональних інструментів, так як має можливість аналізувати трафік між клієнтом та сервером, перевіряти та знаходити можливості експлуатації більшості відомих вразливостей. Широко використовується провідними фахівцями у сфері аналітики та пентесту [7].

Nessus – програмний комплекс, який розрахований здебільшого для корпоративного застосування, має велику кількість різноманітних сканерів, здатних виявляти CVE та вразливості у конфігураціях у різноманітних системах, включно з мережевими сервісами, базами даних, хостовими пристроями.

Acunetix – інструмент, спеціалізований на автоматичному скануванні веб-додатків. Ключова його особливість в тому, що він здатний окрім пошуку вразливостей, виявляти та запобігати веб-атакам, включно з SQL-ін’єкціями.

OWASP ZAP – безкоштовний інструмент проєкту OWASP, що має відкритий вихідний код для тестування веб-додатків. Має широкий функціонал, пов’язаний з веб-вразливостями та атаками.

Нижче наведено таблицю порівнянь зазначених сканерів вразливостей:

Таблиця 1.3

Таблиця порівнянь сканерів вразливостей

Назва сканеру	Класифікація	Основне призначення	Покриття сканування	Доступність
SQLMap	Вузькоспеціалізований інструмент SQLi з функціями DAST.	Виявлення та автоматизована експлуатація і ексфільтрація даних.	Точково, у веб-додатках	Безкоштовний
Burp Suite	Платформа з функціями DAST, а також проксі-серверу, що перехоплює та аналізує трафік між клієнтом та сервером (Intercept proxy).	Перехоплення трафіку з можлвістю ручного та автоматизованого сканування	Веб-додатки та API функції	Має безкоштовну Community версію

Nessus	Багатофункціональний сканер для мережевих вразливостей, має функції IAST	Виявлення CVE, вразливостей у конфігураційних файлах на хостах або сервісах.	Хостові ОС, мережеві сервіси, бази даних, обмежене покриття веб-додатків	Платний сканер для корпоративних рішень
Acunetix	Інструмент, що має функції DAST	Сканер з автоматизованим управлінням, призначений для сканування веб-додатків та API.		Платний
OWASP ZAP	Сканер з функціями DAST та Intercept proxy	Перехоплення трафіку, з можливістю ручного та автоматизованого сканування веб-додатків		Безкоштовний

Методи захисту від вразливості

Існує велика кількість рішень, які будуть допомагати виявляти атаки та блокувати їх, проте не менш важливим у захисті є застосування виправлень або використання методів, які будуть мінімізувати таку загрозу. Нижче наведено приклади захисту на етапі розробки, або впровадження, веб-додатку [8]:

1. Валідація вхідних даних перед тим, як відправити запит до бази даних. Зокрема, це може бути видалення пробілів, екранування або видалення зайвих та неочікуваних спецсимволів, обмеження на кількість символів вхідних даних.
2. Використання ORM інструментів, які призначені для конвертування коду в SQL-запити та навпаки. Для прикладу у зміні передаються вхідні дані користувача, які згодом передаються через ORM-інструмент у базу даних.
3. Використання шаблоних запитів, де у попердньо заготовлені місця будуть розміщуватися аргументи з правильними значеннями.
4. Обмеження прав користувачів при виконанні SQL-запитів до бази даних, для запобігання несанкціонованому доступу до системи. Дотримуватися принципу мінімально необхідних привілеїв.
5. Відключити всі опції, які будуть повертати користувачеві повідомлення про помилку під час обробки запиту, або помилки іншшого типу. При необхідності такі записи можна зберігати в окремій папці з обмеженим доступом.

1.4 Аналіз системи захисту веб-додатків

Існують системи, які виявляють та нейтралізують спроби атаки зловмисників. Використання подібних рішень є доволі ефективним разом із дотримання основних принципів безпеки коду, мінімізації привілегій та ін., проте ми розглянемо систему захисту WAF, зокрема один із продуктів систем такого типу буде впроваджено у межах цієї роботи.

Web Application Firewall (WAF) – це одна із основних систем для захисту веб-додатків, захисний екран, який працює на прикладному рівні (рівні застосунків). WAF здійснює спеціалізований захист веб-додатків, методом аналізу і фільтрування вхідного та вихідного трафіку, шукаючи підозрілу і аномальну активність, або на наявність шкідливих запитів [9].

Як і традиційні мережеві фаєрволи, WAF відповідно до безпекових політик може блокувати, пропускати чи обмежувати шкідливі з’єднання або запити. Зокрема, він здатний виявляти [9]:

1. SQL-ін’єкції.
2. Cross-Site Scripting.
3. DDoS-атаки.
4. Вразливості нульового дня (Zero-Day vulnerabilities).
5. HTTP Exploits.
6. API Exploits.
7. Command Injection.
8. RCE/RFE-атаки.

В залежності від потреб, існує декілька способів інтеграції WAF у інфраструктуру організації [9]:

Хмарний WAF – фаєрвол розміщується у зовнішнього провайдера, та одночасно виконувати функції CDN (мережа доставки контенту). Використання даного рішення дозволяє легко масштабуватися та мінімізувати витрати на обладнання.

Локальний WAF – фаєрвол, який локально розміщений на серверах, що дає повний контроль над налаштуванням, проте вимагає обслуговування.

Гібридний WAF – фаєрвол, що поєднує локальний та хмарне рішення. Для прикладу, коли необхідно зберігати незалежність або постійне функціонування пріоритетного трафіку, використовується локальне рішення, а для менш пріоритетного трафіку – хмарне.

ModSecurity

Розглянемо детальніше систему, яка буде впроваджуватися у даній роботі, як рішення захисту від SQL-ін’єкцій, а саме – ModSecurity.

ModSecurity – це один із найпопулярніших фаєрволів для веб-додатків (WAF), що працює з веб-серверами Apache, Nginx та IIS. Аналіз трафіку відбувається на основі правил та алгоритмів виявлення загроз, що пропускає лише легітимний трафік, відсіюючи шкідливий та небезпечний. Принцип аналізу наступний:

- WAF аналізує кожен запит та відповідь від сервера на основі патернів в пошуку аномалій;
- далі цей трафік порівнюється з політиками безпеки, які були налаштовані користувачем, базою загроз та іншими правилами;
- у разі виявлення шкідливого трафіку, фаєрвол блокує або детектує, якщо у політиках безпеки було дозволено пропускати всі запити.

Працювати WAF може за двома моделями: Negative Security Model або Positive Security Model. Це основні принципи функціонування захисту, де дозволено весь трафік, окрім виключень, або блокується весь трафік, окрім виключень.

ModSecurity являється посередником між клієнтом та веб-сервером, захищаючи базу даних та інші важливий функціонал серверу. Його розташування в у комп'ютерній системі веб-додатку можна побачити на рисунку 1:

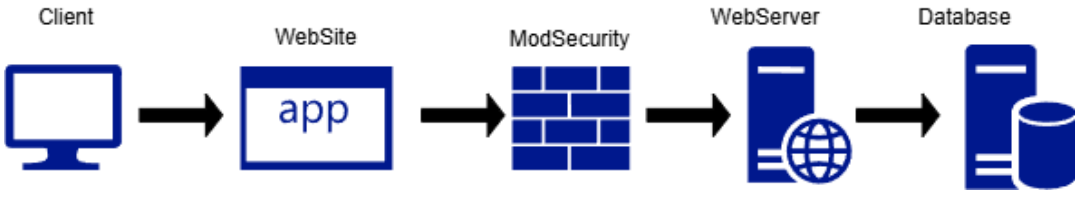


Рис. 1.1. Схема взаємодії мій клієнтом та сервером через WAF

Є перелік ключових особливостей ModSecurity, які заставляють зупинитися на ньому, у виборі між різними системами веб-захисту. Окрім виявлення атак та їх блокування у HTTP-трафіку, фаєрвол здатний збирати деталізовані журнали запитів, передавати їх, що дає змогу моніторити події в реальному часі. Завдяки цьому функціоналу, ModSecurity часто комбінують з різними SIEM-системами,

такими як Splunk або ELK Stack (ElasticSearch + Kibana), що допоможе в ході дослідження атак з використанням SQL-ін’єкцій.

Також, ModSecurity має ще дві важливі переваги:

- Використання WAF підвищує загальну безпеку веб-додатку, навіть при умові вразливості, бо ModSecurity блокуватиме атаку на рівні трафіку, перш ніж вона дійде до веб-серверу.
- ModSecurity має набір правил, інтегрованих з OWASP Core Rule Set (CRS), які охоплюють більшість відомих атак згідно з OWASP Top 10. Це дозволяє швидко і ефективно розгорнути захист, не витрачаючи багато часу на конфігурування та створенная політик безпеки [10].

1.5 Аналіз системи моніторингу у виявлені SQL-ін’єкцій

При масштабувані інфраструктури, поверхня для атаки зловмисників зростає, що змушує підходити до питання безпеки відповідально, враховуючи так же ефективність впровадження рішень. Мережевий, або фаєрвол веб-додатків не дає сто відстокового захисту, а атаки та спроби експлуатації вразливостей необхідно виявляти для вчасного та швидкого реагування, зменшуючи тим самим ризик компрометації системи.

Серед різноманіття пристроїв, з різним призначенням, будовою та функціоналом, у всіх є спільна риса – записи журналів подій. Саме ці записи, або як їх ще називають – логи, дозволяють фахівцю з кіберзпеки, системному адміністратору бачити і розуміти всю «картину» подій. І з моменту розвитку кіберзагроз, відповідно розвивалися рішення безпеки, де одна з них – це SIEM-додатки.

SIEM-система (Security Information and Event Management) – це програмне рішення, яке централізовано збирає та корелює журнали подій безпеки з усієї інфраструктури в реальному часі. Дане рішення необхідне для проактивного

виявлення кіберзагроз та обізнаності щодо стану безпеки організації, а головна мета – це скоротити час між виникненням інциденту та його виявленням (MTTD) і реагуванням (MTTR) [11].

По своїй суті SIEM-системи об’єднують пасивний збір логів (Log Management) разом із активною кореляцією подій в режимі реального часу, а кореляція може здійснюватися на основі вбудованих, так і користувацьких правил. Для прикладу можна створити правило на кількість спроб з’єднань з однієї IP-адреси за короткий проміжок часу, і при спрацюванні, повідомляти оператора про підозрілу подію.

Однак, існують інші рішення для моніторингу, а саме XDR та EDR системи. SIEM-системи в першу чергу призначені для збору, нормалізації та аналізу логів, проте XDR та EDR системи розроблені для проактивного виявлення загроз.

EDR-система (Endpoint Detection and Response) – це система моніторингу кінцевих точок користувачів, та підходить здебільшого для виявлення та блокування шкідливих програм, файлів та подій на пристрої. Загалом EDR є гарним рішенням захисту, проте в даному дослідженні він не покриває необхідних вимог. Саме тому слід звернути увагу на XDR-системи.

XDR-система (Extended Detection and Response) – це система розширеного моніторингу інфраструктури, що покриває не тільки захист кінцевих точок, а ще й аналізує мережевий трафік, поведінку користувачів, хмарне середовище, дані, що зберігаються на пристрої.

У виборі ефективного рішення для захисту, а також виявлення загроз слід обирати XDR-системи, оскільки SIEM не можуть автоматично реагувати на загрози, а також більшість з них мають необхідність налаштування та витрат доволі великого об’єму часу для аналізу журналів подій. В таблиці 1.4 наведено коротке порівняння

SIEM та XDR, а також ключові фактори, які повпливали на вибір у межах дослідження атаки з використанням SQL-ін’єкції [12]:

Таблиця 1.4

Таблиця порівнянь SIEM та XDR

SIEM	XDR
Призначений для аналізу журналів подій, що збираються в організації.	Призначений для автоматичного виявлення загроз та їх блокування.
Система потребує ручного обслуговування, створення/написання правил детектування.	Постачальник системи автоматично оновляє базу даних загроз для виявлення та реагування.
SIEM системи сильно програють по часу реагування, так як час затрачений на аналіз є доволі великим.	Автоматизація більшості процесів дозволяє ефективно блокування шкідливі процеси або запити.

SIEM є доволі корисним інструментом, проте ізольованість під час збору даних, а також великі обсяги збору журналів ставлять під сумнів ефективність захисту. Саме тому, в ході аналізу систем моніторингу подій, було вирішено зупинитися на XDR-системі.

XDR-система WAZUH

Wazuh – це платформа з відкритим вихідним кодом, що поєднує одночасно можливості XDR та SIEM. Основне призначення це захист від кіберзагроз, а також можливості моніторингу безпеки, розширене виявлення загроз, моніторинг цілісності файлів, пошук на відомі вразливості, хмарний захист та багато іншого [13].

Коротка збірка можливостей платформи Wazuh:

- виявлення вторгнень IDS/IPS;
- автоматична кореляція подій;
- можливість аналізу журналів подій;
- управління вразливостями кінцевих точок та пристроїв;
- розвідка загроз та їх кореляція;
- можливість легкого масштабування.

В сукупності, рішення використовувати Wazuh як відкриту та безкоштовну платформу є найбільш оптимальним серед існуючих продуктів на ринку, а детально проаналізувати роботу цієї XDR-системи можна буде в наступних розділах.

1.6 Постановка задачі до виконання

Дане дослідження орієнтоване на зменшення загрози атак з використанням вразливостей SQL-ін’єкцій, та підвищення рівня безпеки в системі. Для досягнення успішного результату цього дослідження необхідно чітко сформуваит перелік задач, які мають бути виконанні:

1. Побудувати комп’ютерну систему з вразливим веб-додатком до атак SQL-ін’єкцій.
2. Проаналізувати існуючі методи захисту від веб-атак з SQLi.
3. Забезпечити ізольованість мережі в ході дослідження, а також впровадити системи моніторингу для детектування атаки.
4. Провести тестування системи в ході атаки з використанням вразливостей додатку.
5. Впровадити системи захисту (фаєрвол веб-додатків) та перевірити корректність роботи.
6. Провести аналіз результатів дослідження та задокументувати їх.

В підсумку розділу було розглянуто що таке SQL-ін’єкції, які існують методи виявлення та захисту, а також обрано наступні системи для використання:

- У якості рішення для захисту від веб-атак було вибрано WAF ModSecurity.
- Для моніторингу та виявлення загроз було обрано XDR-систему Wazuh.
- Для проведення тестування на вразливість було обрано платформи Burp Suite та SQLMap.

РОЗДІЛ 2. РОЗРОБКА ВЕБ-ДОДАТКУ ТА ВПРОВАДЖЕННЯ СИСТЕМИ МОНІТОРИНГУ

2.1 Опис архітектури КС

Першочерговим етапом при дослідженні атаки є побудова необхідних лабораторних умов, а також створення комп'ютерної системи, на яку буде здійснювати атака. Комп'ютерна система складається з серверної інфраструктури, веб-додатку, а також впровадження системи моніторингу журналів подій безпеки, що дозволить виявляти здійснені атаки на КС та підвищити рівень безпеки інформаційної системи.

Під час процесу розгортання та інтеграції серверної інфраструкри слід досконально вивчити такі аспекти як машстабованість, безпека та ефективність в управлінні ресурсами. Саме тому, для серверного середовища було обрано Linux-дистрибутив, а саме Ubuntu Server, що включає:

- відкритий вихідний код;
- масштабованість;
- гнучкість налаштувань;
- підтримка великої кількості серверних додатків;
- розвинута система безпеки та управління.

Ці аспекти роблять вибір Ubuntu одним з найкращих для побудови серверної інфраструктури, що признано багатьма системними адміністраторами та розробниками у світі. Операційна система Linux, а саме дистрибутив Ubuntu буде розгорнено у віртуальному середовищі VMware, що дає змогу побудувати віртуальне середовище без необхідності мати велику кількість пристроїв.

Структура серверної інфраструктури:

Структура на базі Ubuntu включає такі основні компоненти:

Веб-сервер на базі Nginx, що забезпечує обробку HTTP-запитів, та відображає веб-додаток. На відміну від багатьох веб-серверів, які обробляють запити по черзі, Nginx відомий свою асинхронною архітектурою, можливістю керуванням подій, що дозволяє ефективно та швидко обробляти велику кількість з'єднань одночасно з мінімальним споживанням ресурсів [14].

Система управління базами даних (СУБД), а саме було вибрано PostgreSQL – широко розповсюджену систему з відкритим кодом, що являється альтернативою для комерційних СУБД як Oracle Database, IBM DB2 або MS SQL Server. Ця база даних була обрана через масштабованість і гнучкість, що дозволяє працювати з великими обсягами даних та легкого інтегрування з backend-додатками. В якості інструмента для взаємодії з СУБД було обрано інструмент управління з відкритим вихідним кодом pgAdmin.

Модулі безпеки AppArmor [15] для контролю доступу на основі профілю процесів, що визначає, до яких ресурсів окремий процес має доступ, а також модуль UFW (Uncomplicated Firewall) [16], що являється мережевим фаєрволом.

Також, в комплекс серверної інфраструктури входить Wazuh Server, який включає наступні компоненти [17]:

- Wazuh Manager, що являється основним сервером для збору та аналізу журналів подій;
- Wazuh Agent – клієнтський агент, що встановлюється на кінцевих точках та пристроях.
- Wazuh Indexer – вбудоване сховище зберігання логів, що використовується для подальшої аналітики.
- Filebeat – агент для збору та доставки журналів подій.

Нижче, на рисунку 2.1 зображено схему архітектури потоку даних, які обробляються в ході збору, аналізу та відображення. Також, зазначено опис етапів обробки даних на кожному із ключових компонентів системи.

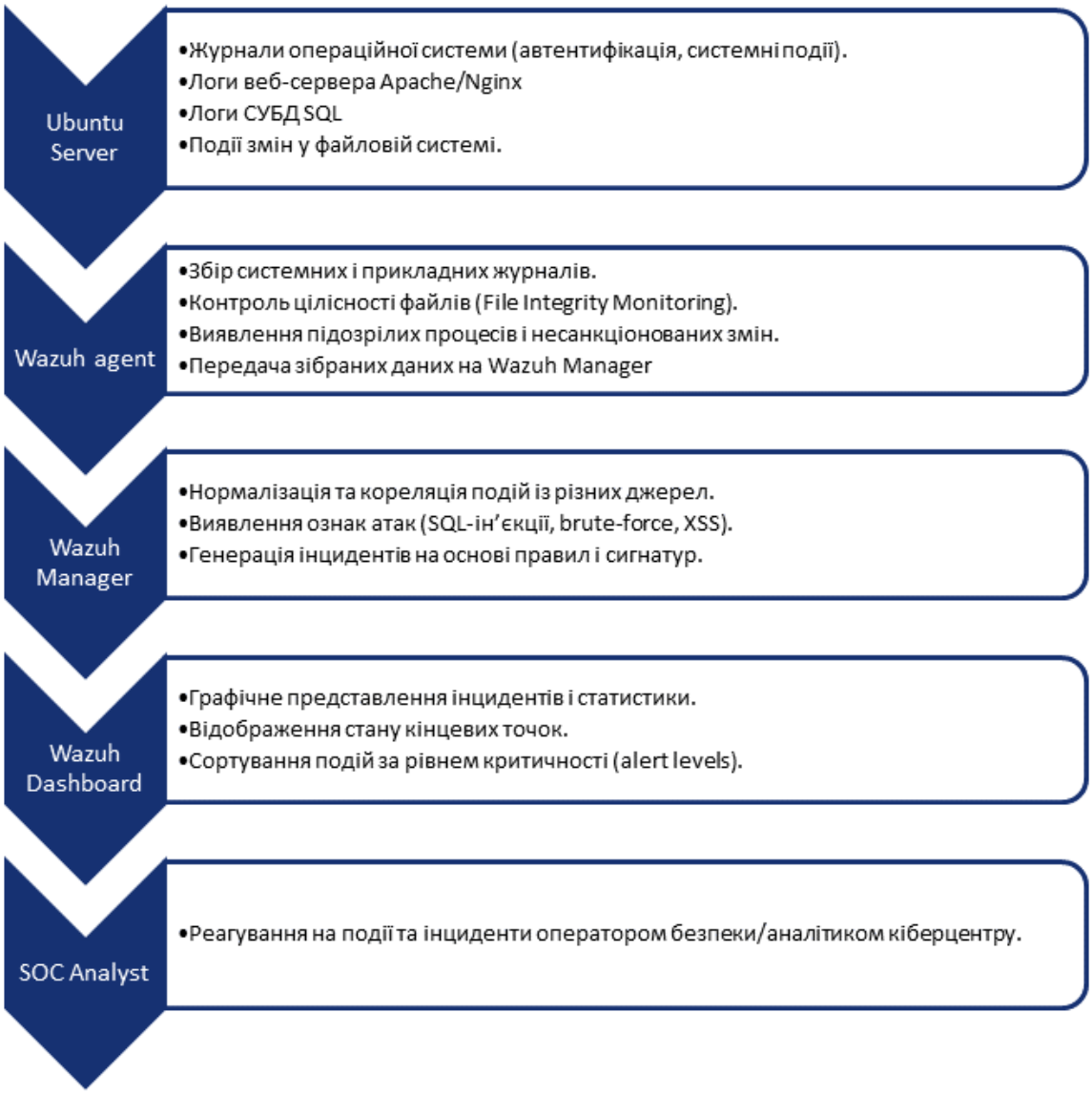


Рис. 2.1. Схема архітектури потоку даних

2.2 Технічні характеристики компонентів КС

В попередньому розділі було зазначено компоненти, які використовувалися для побудови комп'ютерної системи та розгортання серверної інфраструктрури. В

цьому розділі наведено технічні характеристики з коротким описом кожного компоненту:

1. Операційна система серверної інфраструктури:

Для розгортання серверної інфраструктури було обрано ОС Linux з дистрибутовом Ubuntu версії 24.04.02 LTS, що на момент розробки було останньою версією LTS Ubuntu. Тип операційної системи – 64-біти, та побудована на базі ядра Linux 6.8.

Основні характеристики:

- стабільна підтримка релізу;
- підтримка систем юезпеки AppArmor, UFW;
- сумусність з популярними серверними додатками;
- підтримка оновлень безпеки.

Рекомендовані технічні вимоги:

- 4 ядра CPU, 8 ГБ оперативної пам’яті та 50 ГБ віделоної пам’яті на диску.

2. Система управління базами даних PostgreSQL:

СУБД PostgreSQL призначений для зберігання та обробки даних користувачів у веб-додатку, з використанням графічного інтерфейсу адміністрування pgAdmin4.

Основні характеристики:

- система контролю транзакцій ACID;
- розширення безпеки pgcrypto, SSL/TLS та принцип row-level security;
- графічний інтерфейс для адміністрування та моніторингу pgAdmin4.

Рекомендовані технічні вимоги:

- 2 ядра CPU, 4 ГБ оперативної пам’яті, 10 ГБ виділеної пам’яті на диску та відкриті порти для TCP з’єднань (5432 для PostgreSQL та 5050 для pgAdmin).

3. Веб-сервер Nginx

Nginx був обраний як веб-сервер для відображення статичного контенту веб-додатку, обробки HTTP-запитів користувачів та передачі даних до PHP скриптів, які виконують роль backend-частини додатку.

Основні характеристики:

- підтримка протоколів HTTP 1.1, HTTP 2, HTTPS;
- розподіл навантаження та кешування;
- захист від DoS атак за допомогою механізму обмеження запитів;
- можливість інтеграції з PHP-FPM для виконання динамічних сценаріїв у додатку.

Рекомендовані технічні вимоги:

- 1 ядро CPU, 1 ГБ оперативної пам’яті, відкриті порти TCP 80 та 443.

4. Віртуалізоване середовище VMware Workstation 17

VMware використовується для створення ізольованого лабораторного середовища, що дає можливість безпечно дослідити веб-атаку.

Основні характеристики:

- підтримка мережевих адаптерів NAT та Bridged;
- можливість створення снепшотів (знімків стану системи) для відновлення;
- підтримка віртуалізації апаратних ресурсів.

Рекомендовані технічні вимоги [18]:

- 4 ядра CPU, 16 ГБ оперативної пам’яті та 256 ГБ вільної пам’яті на диску.

5. Платформа моніторингу безпеки Wazuh

Для моніторингу кінцевої точки та виявлення атак було розгорнуто центральну систему моніторингу безпеки серверного середовища Wazuh Server, що включає такі компоненти як Wazuh Manager та Wazuh Dashboard.

Основні характеристики:

- виявлення вторгнень HIDS;
- моніторинг цілісності файлів FIM;
- аудит системи Auditd;
- інтеграція з Elasticsearch, Kibana та REST API;
- моніторинг та аналіз подій в реальному часі.

Рекомендовані технічні вимоги [19]:

- 4 ядра CPU, 8 ГБ оперативної пам’яті, 100 ГБ вільної пам’яті на диску, відкриті порти для TCP з’єднань (55000 для зв’язку з Wazuh Agent, 443 для Wazuh Dashboard).

2.3 Розробка веб-додатку та інтеграція з базою даних SQL

В цьому підрозділі описано процес розробки веб-додатку, а також розгортання бази даних SQL, та інтеграція з додатком.

Розробка веб-додатку

Веб-додаток – це програма, яку можна відкрити за допомогою будь-якого браузера. Веб-додаток, що розробляється, складається з клієнтської (frontend) та серверної (backend) частин.

Зовнішній інтерфейс веб-додатку (frontend) розроблявся за допомогою таких мов програмування як: HTML, CSS, Javascript, які підтримуються будь-яким браузером (Edge, Chrome, Mozilla, Brave, тощо).

Додаток має наступний список файлів, які створюють візуальний інтерактив та використовуються для роботи сценаріїв:

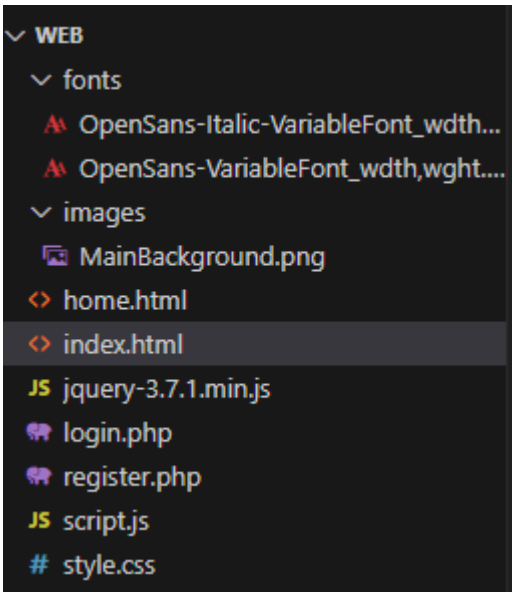


Рис.2.2. Структура проєкту веб-додатку

index.html – це файл головної сторінки веб-сайту, де було розроблено основні елементи та наповнення. На сторінці присутні кнопки входу та реєстрації, які створені у вигляді віджетів. Віджети виконанні у вигляді двох окремих контейнерів класу «form-box login» та «form-box register». Переглянути html-код сторінки можна у додатку А.1.

Дані контейнери зазвичай приховані від користувача, і стають доступні лише при натисканні кнопки «Login» чи «Register». Кожен із контейнерів має свої рядки для вводу даних, проте ключова різниця полягає в передачі даних з використанням методу «POST»:

<form class="login-form" action="login.php" method="POST">

<form class="register-form" action="register.php" method="POST">

login-form перенаправляє введення дані через POST-запит в login.php, а register-form перенапрявляє у register.php, що дозволяє зберегти дані. HTML не дає змогу напряму

взаємодіяти із backend частиною веб-сайту, тому було прийнято рішення використовувати JavaScript.

home.html – використовується як наступна сторінка під час входу користувача, та відображається у разі успішного входу. Ніяких функцій та цілей, окрім візуального відображення ця сторінка не має.

style.css – файл візуальних та стильових налаштувань елементів на веб-сторінках.

login.php та register.php – використовуються для взаємодії з сервером та базою даних, зокрема перевірки облікових даних користувача під час входу, або реєстрація нового користувача в базі даних. Код PHP-скриптів login.php та register.php представлено у додатках А.2 та А.3 відповідно.

script.js – файл скриптів, що виконуються при певних заданих умовах. В цьому файлі присутні лише прості дії, що формують візуальну частину сайту, а саме блюр фону при натисканні кнопки «Login».

Jquery-3.7.1.min.js - бібліотека спрощеного управління сприптами, а також для використання в роботі Ajax – асинхронної взаємодії веб-додатку з сервером та базою даних [20, 21]. Найбільш цікавим для вивчення є механізм роботи Ajax, методу дозволяє передавати дані до login.php та серіалізувати їх, а також робити перевірки, чи введені дані збігаються з тими, що наявні в базі даних за допомогою register.php.

На рисунку 2.3 можна ознайомитися з спрощеною схемою логіки роботи веб-додатку зі сторони користувача:

Рис. 2.3. Логічна схема веб-додатку

Варто зауважити, що даний додаток не побудований для використання у комерційних або інших цілях, окрім як демонстрації проведення веб-атаки з використанням вразливості SQL-ін’єкції. Головну сторінку веб-додатку з віджетом «Login» можна побачити на рисунку 2.4:

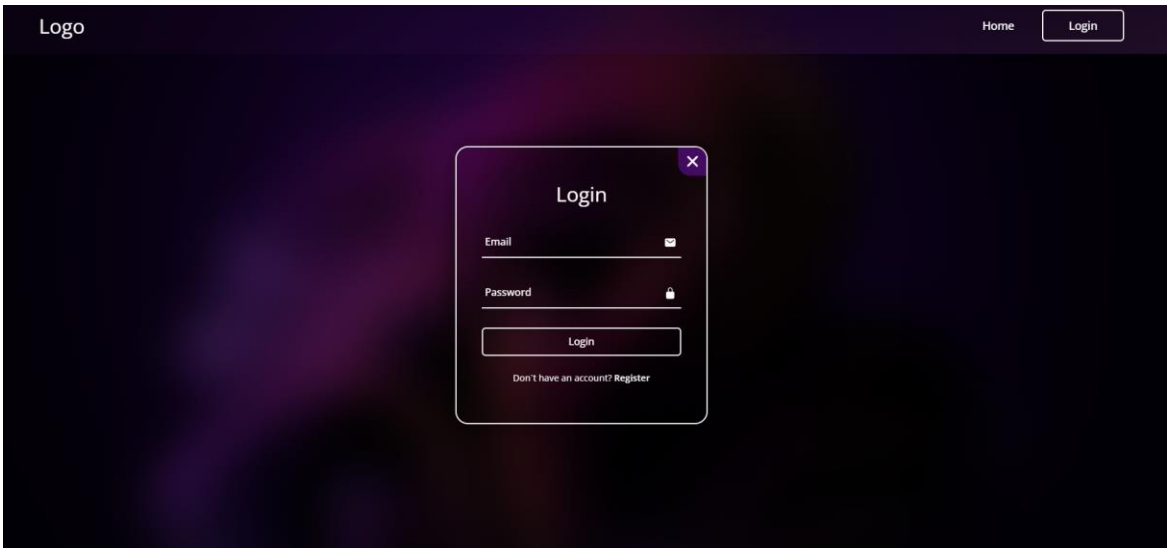


Рис. 2.4. Головна сторінка веб-додатку

Розгортання бази даних SQL

Встановлення СУБД PostgreSQL на сервері Ubuntu Server є доволі легким за допомогою пакетного менеджеру apt, що можна зробити одним рядком команди:

sudo apt install postgresql

Проте, використання консольної версії бази даних не є зручним та ефективним, зокрема через велику кількість команд, які необхідно використовувати, а також у інтерфейсі відображення, який ускладнює пошук баз даних або користувачів. Для вирішення цієї проблеми вибрано пакет pgAdmin4, що є безкоштовним графічним інтерфейсом з відкритим вихідним кодом, та розробений спеціально для адміністрування баз даних. Встановлення відбувається за допомогою переліку команд:

Створення ключа-сертифікату для можливості з’єднання з репозиторієм:

curl -fsS https://www.pgadmin.org/static/packages_pgadmin_org.pub | sudo gpg --dearmor -o /usr/share/keyrings/packages-pgadmin-org.gpg

З’єднання з репозиторієм PostgreSQL для завантаження інструменту pgAdmin:

sudo sh -c 'echo "deb [signed-by=/usr/share/keyrings/packages-pgadmin-org.gpg] https://ftp.postgresql.org/pub/pgadmin/pgadmin4/apt/$(lsb_release -cs) pgadmin4 main" > /etc/apt/sources.list.d/pgadmin4.list && apt update'

Після з’єднання у користувача з’являється можливість завантаження десктопної або веб-версії інструменту:

sudo apt install pgadmin4-desktop

Після завантаження pgAdmin отримується можливість створення root-користувача, а також адрес та порт на яких буде працювати база даних. В даній інфраструктурі база даних розміщується в одній системі з веб-сервером, так як

можливість масштабування не є важливою в цьому дослідженні. Отже, налаштування бази даних має наступний вигляд:

- Host: localhost;
- Port: 5432;
- Username: postgres.

Після створення з'єднання, за допомогою інструмента pgAdmin було створено базу даних «WebsiteDB» та таблицю «users». Структуру таблиці та вигляд створеної бази даних можна побачити на рисунку 2.5:

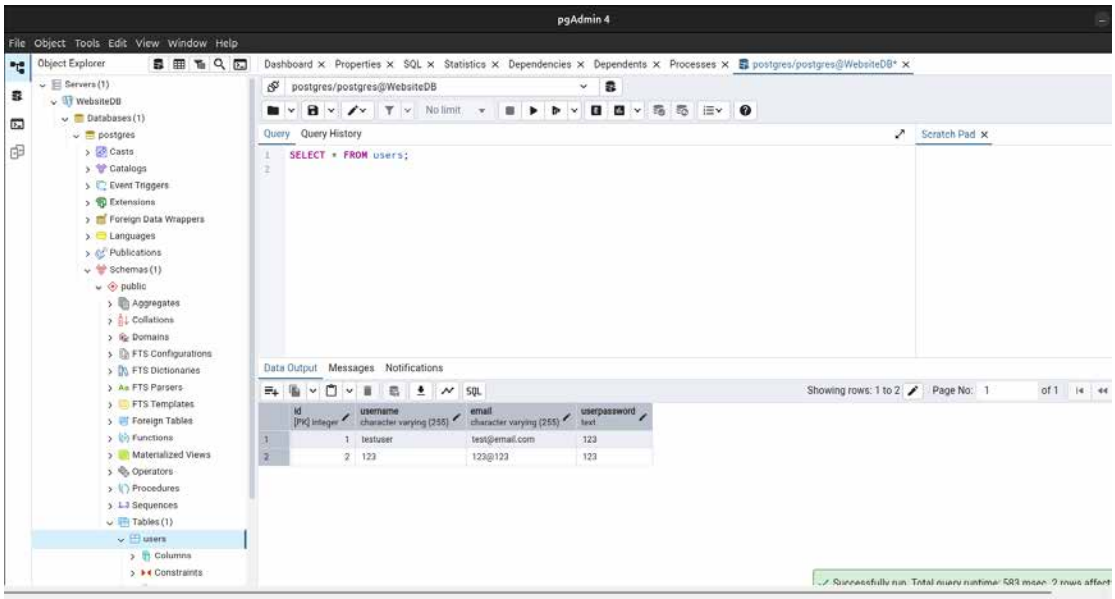


Рис. 2.5. Вигляд та структура бази даних «WebsiteDB»

Розробка серверної частини веб-додатку та інтеграція з базою даних

Backend частина веб-додатку відповідає за обробку запитів клієнтів, управління базою даних та забезпечення безпеки взаємодії користувачів із системою. Серверна частина веб-додатку була реалізована з використанням мови програмування PHP, на якому було написано сценарії з'єднання та взаємодії з базою даних.

Існує три основних сценарії:

- створення з’єднання з базою даних;
- автентифікація існуючого користувача;
- реєстрація нового користувача.

З’єднання з базою даних відбувається через параметри підключення, які були зазначені в попередньому підрозділі (host, port, dbname, username, password), а також PHP-функцією pg_connect().

Сценарій автентифікації користувача перевіряє введені користувачем дані та надає доступ при наявному записі у базі даних. Схема алгоритму наведено на рисунку 2.6.

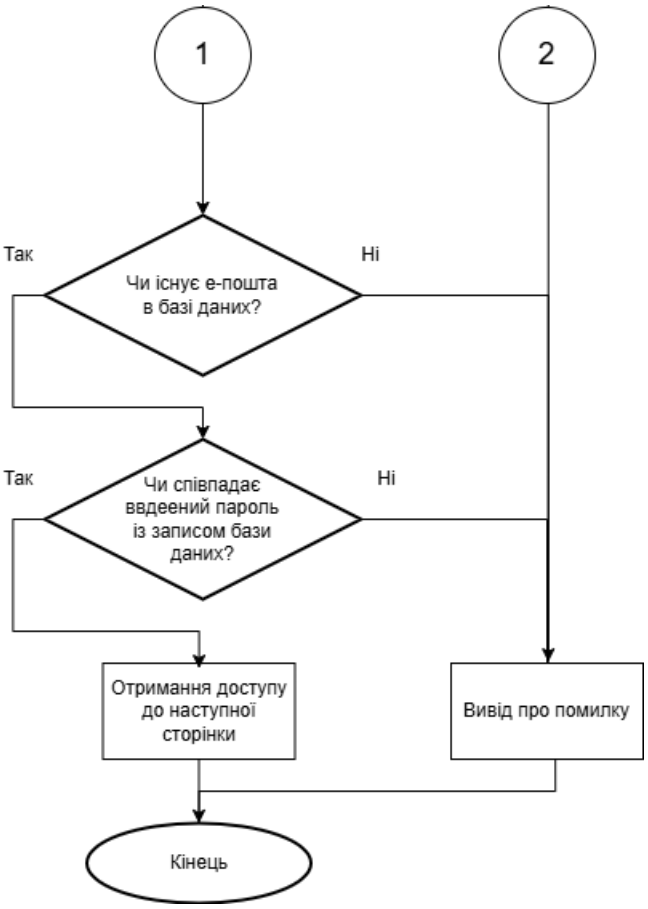
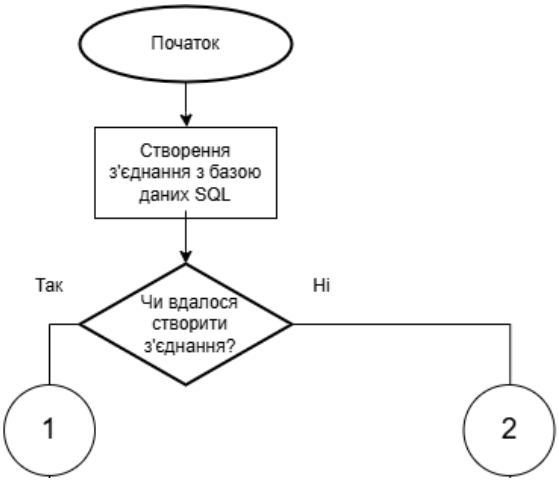


Рис.2.6. Узагальнений алгоритм сценарію автентифікації

Другим сценарієм є реєстрація корисутвачів у веб-додатку, що означає збереження даних про користувача у базі даних, забезпечуючи також цілісність введених даних. На рисунку 2.7 наведено алгоритм реєстрації користувача та внесення даних у базу даних SQL:



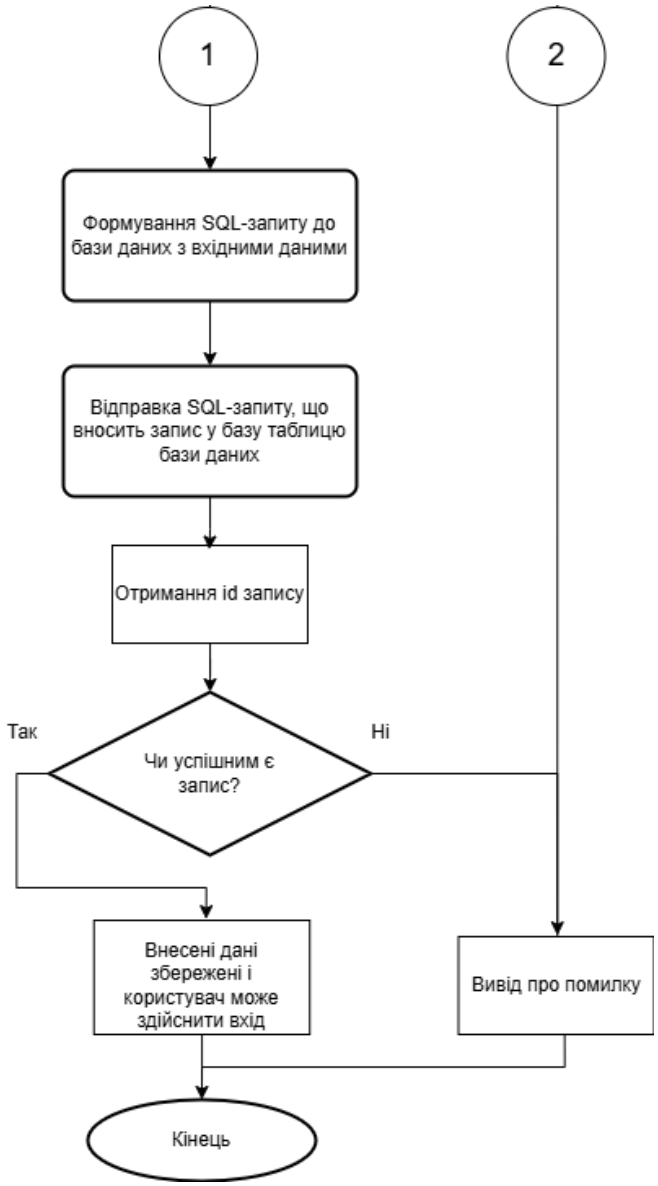


Рис. 2.7. Узагальнений алгоритм сценарію реєстрації

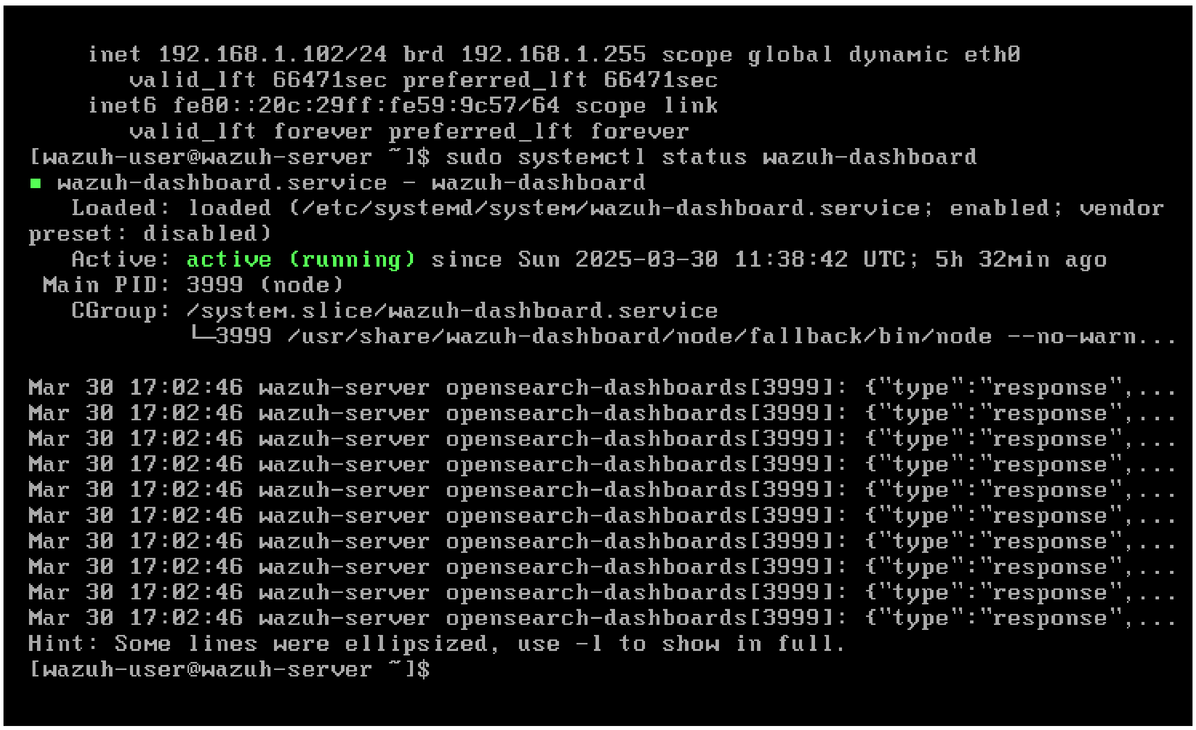
Ознайомитися з PHP-кодом додатку можна у додатках А.2 та А.3.

2.4 Розгортання та впровадження системи моніторингу та пошуку загроз Wazuh

Розгортання відбувається у віртуальному середовищі, і для зручності роботи існує virtual appliance (OVA-файл) для швидкого розгортування системи [19]. Система Wazuh побудована на дистрибутиві Amazon Linux 2023, та включає наступні центральні компоненти:

- Wazuh manager 4.11.1
- Filebeat-OSS 7.10.2
- Wazuh indexer 4.11.1
- Wazuh dashboard 4.11.1

Доступ до даної системи можна отримати через ssh клієнт, або через веб-інтерфейс користувача (Wazuh Dashboard). Перевірити роботу Wazuh Dashboard можна в терміналі Wazuh Manager, як показано на рисунку 2.8:



```
    inet 192.168.1.102/24 brd 192.168.1.255 scope global dynamic eth0
       valid_lft 66471sec preferred_lft 66471sec
    inet6 fe80::20c:29ff:fe59:9c57/64 scope link
       valid_lft forever preferred_lft forever
[wazuh-user@wazuh-server ~]$ sudo systemctl status wazuh-dashboard
■ wazuh-dashboard.service - wazuh-dashboard
   Loaded: loaded (/etc/systemd/system/wazuh-dashboard.service; enabled; vendor
preset: disabled)
   Active: active (running) since Sun 2025-03-30 11:38:42 UTC; 5h 32min ago
 Main PID: 3999 (node)
   CGroup: /system.slice/wazuh-dashboard.service
           └─3999 /usr/share/wazuh-dashboard/node/fallback/bin/node --no-warn...

Mar 30 17:02:46 wazuh-server opensearch-dashboards[3999]: {"type":"response",...
Mar 30 17:02:46 wazuh-server opensearch-dashboards[3999]: {"type":"response",...
Mar 30 17:02:46 wazuh-server opensearch-dashboards[3999]: {"type":"response",...
Mar 30 17:02:46 wazuh-server opensearch-dashboards[3999]: {"type":"response",...
Mar 30 17:02:46 wazuh-server opensearch-dashboards[3999]: {"type":"response",...
Mar 30 17:02:46 wazuh-server opensearch-dashboards[3999]: {"type":"response",...
Mar 30 17:02:46 wazuh-server opensearch-dashboards[3999]: {"type":"response",...
Mar 30 17:02:46 wazuh-server opensearch-dashboards[3999]: {"type":"response",...
Mar 30 17:02:46 wazuh-server opensearch-dashboards[3999]: {"type":"response",...
Mar 30 17:02:46 wazuh-server opensearch-dashboards[3999]: {"type":"response",...
Hint: Some lines were ellipsized, use -l to show in full.
[wazuh-user@wazuh-server ~]$
```

Рис. 2.8. Приклад успішного запуску компоненту системи

Переконавшись у правильності роботи середовища, слід підключитися до веб-інтерфейсу Wazuh, який знаходиться за адресом, на якому розміщено Wazuh Manager. Здійснити це можна за допомогою будь-якого браузеру, де перед користувачем з’явиться головна сторінка, яка зображено на рисунку 2.9:

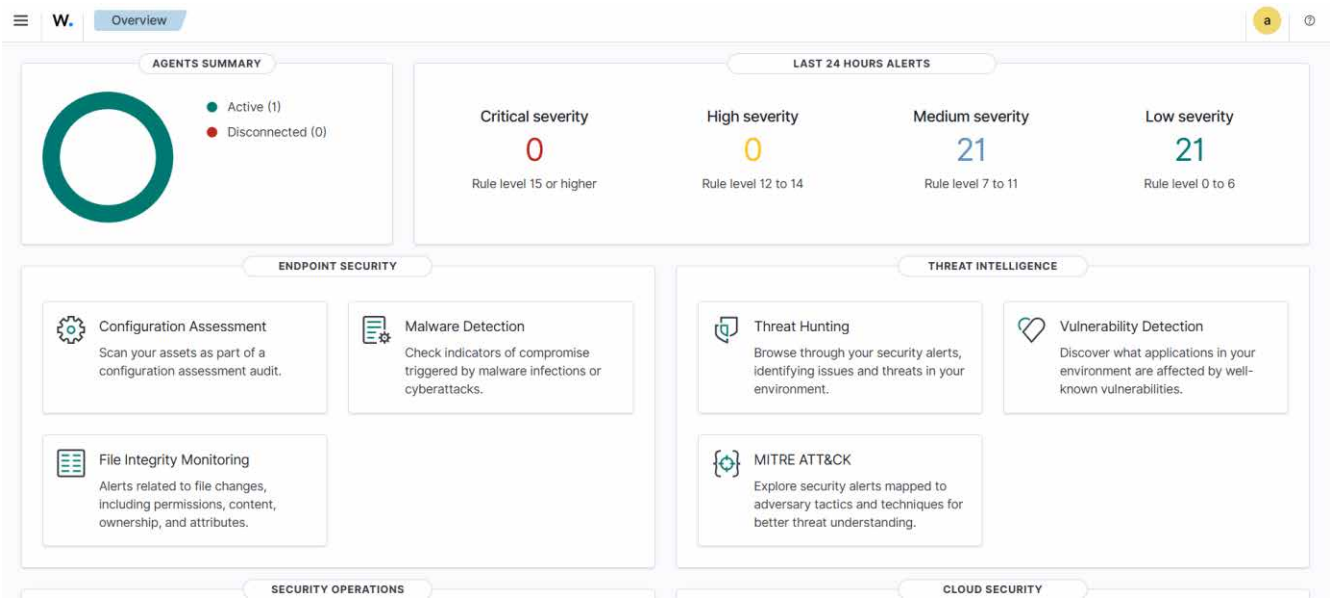


Рис. 2.9. Веб-інтерфейс Wazuh

Встановлення агенту на кінцеву точку

Встановлення агенту може здійснюватися двома способами:

Перший – через панель Endpoints у Wazuh можна обрати встановлення агенту «Deploy new agent», де потрібно вказати тип ОС, на який встановлюється агент, а також адрес серверу. Wazuh сам сформує командний рядок, який зв'яжеться з репозиторієм packages.wazuh.com та автоматично встановиться і запуститься на агенті.

Другий спосіб зазвичай використовується при виникненні помилок, та потребує вручну встановити GPG-ключ на хост, добавити репозиторій packages.wazuh.com до APT та встановити агент через командний рядок, вказавши адрес Wazuh Manager [22].

Після цього, сервер Ubuntu має з'явитися у списку агентів Wazuh Dashboard, як показано на рисунку 2.10:

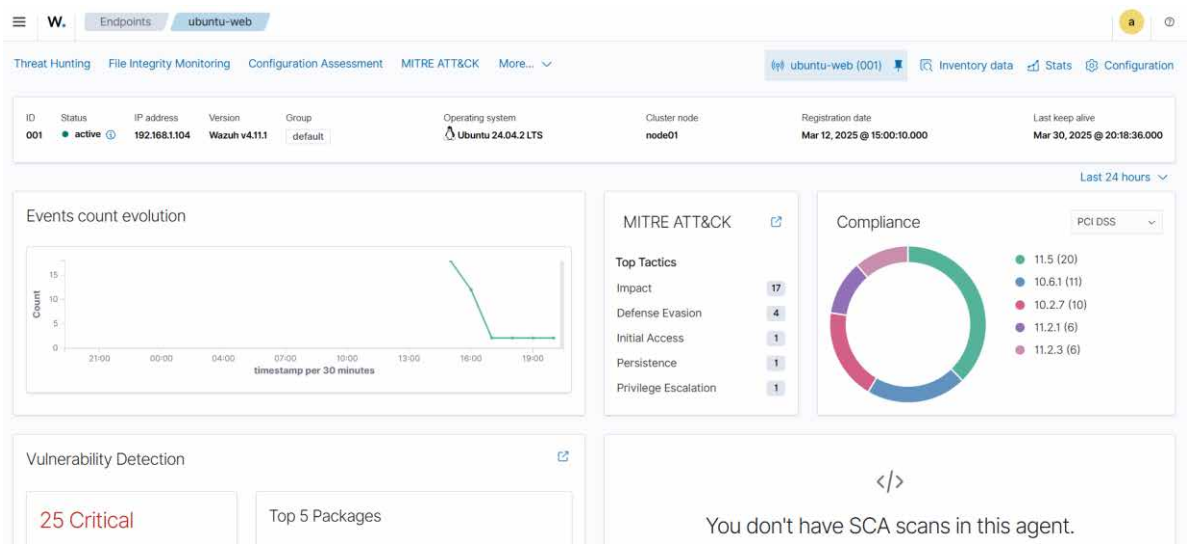


Рис. 2.10. Огляд агенту серверу у Wazuh Dashboard

2.5 Впровадження механізмів безпеки для дослідження атаки

Дане дослідження проводиться для визначення ризиків атаки з використанням SQL-ін’єкції, мета якого мінімізація загрози, забезпечення цілісність експериментальної інфраструктури, а також збір даних телеметрії, для пост-аналізу подій. Для зазначених дій слід дотримуватися безпеки тестового середовища, зокрема в межах дослідження було впроваджено:

1. Ізоляція мережі для тестування від основної, що формує окрему тестову зону для дослідження та комунікації.
2. Створення резервних копій (холодних бекапів) та знімків стану системи (snapshots) віртуального середовища, що легко може повернути систему до функціонального стану.
3. Реалізовано розширений збір подій входу у веб-додатку, для виявлення аномальних чи підозрілих з’єднань. Реалізовано це за допомогоб файлу конфігурацій Nginx, який знаходиться в теці «/etc/nginx/nginx.conf». Приклад формату логу, що збирається системою:

   log_format custom '$remote_addr - $remote_user [$time_local] "$request" '

```
'$status $body_bytes_sent "$http_referer" '
'"$http_user_agent" "$http_x_forwarded_for" '
'$request_time $upstream_response_time '
'$http_cookie $request_method $uri '
'$query_string $document_root';
access_log /var/log/nginx/access_custom.log custom;
```

4. Централізований збір телеметрії та подій на кінцевій точці, а також відстеження цілісності змін у файлах всієї системи. Це було реалізовано в підрозділі 2.4, а також в рамках зменшення часу реагування, у файлі конфігурацій, що знаходиться в теці «/var/ossec/etc/ossec.conf», збір логів було налаштовано на кожні 10 секунд часу:

   <frequency>10</frequency>

   На рисунку 2.11 зазначено директорії, що моніторяться системою, а також частота збору логів на кінцевій точці:

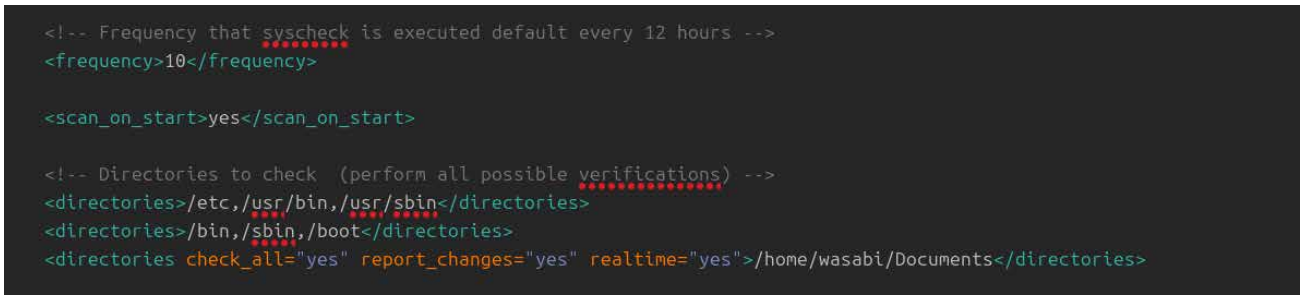


Рис. 2.11. Зображення конфігураційного файлу моніторингу Wazuh

5. Віртуалізація тестового середовища, що ізолює (локалізує) систему від основної для створення необхідних лабораторних умов.
6. Сегрегація даних, що мінімізує ризики під час імітації витоку даних.

РОЗДІЛ 3. ДОСЛІДЖЕННЯ АТАКИ ТА АНАЛІЗ ЗАСОБІВ ЗАХИСТУ

3.1 Дослідження SQL-ін’єкції та виявлення атаки засобами Wazuh

Метою цього підрозділу є проведення атаки з використанням SQL-ін’єкції, щоб показати дану вразливість зі сторони зловмисника, а також область детектування XDR Wazuh. Першочергово здійснюється розвідка середовища, а саме веб-сторінки. При огляді сторінки в панелі розробника (F12 / Dev Tools), що зображено на рисунку 3.1, ми бачимо файли html, css та javascript. Проте в index.html видно, що введені дані з форми, сторінка відправляє до бекенд-частини сайту, що розміщується у файлах login.php та register.php, що може вказувати на обробку та збереження даних сервером.

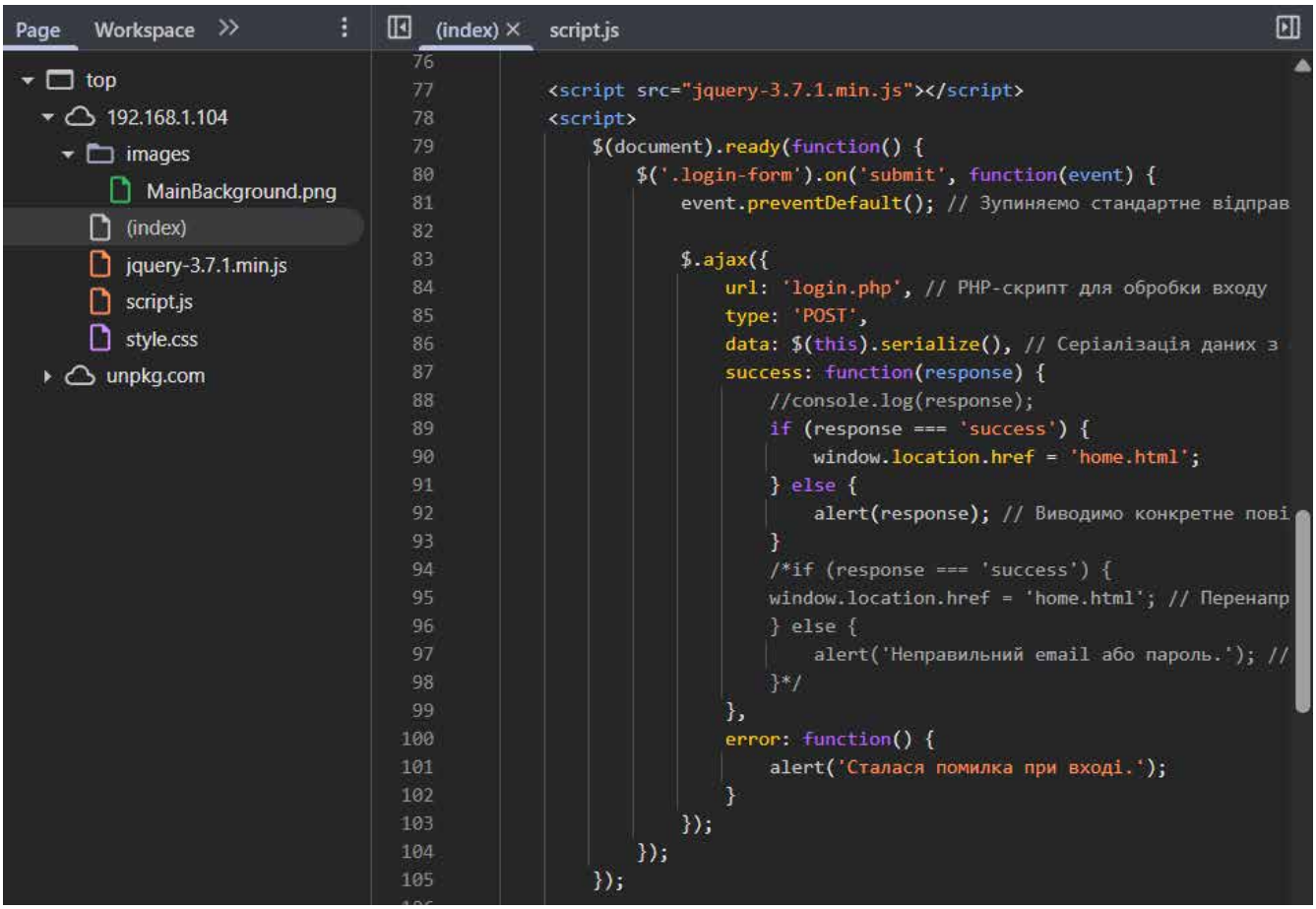


Рис. 3.1. Проєкт веб-сторінки у Dev Tools

Використовуючи платформу для пошуку вразливостей Burp Suite, ми відправляємо довільні дані форми на веб-сервер, та бачимо формат обробки введених даних. На рисунку 3.2 зображено формат відправленого запиту, де варто виділити поля, які можна замінити для сканування на вразливість:

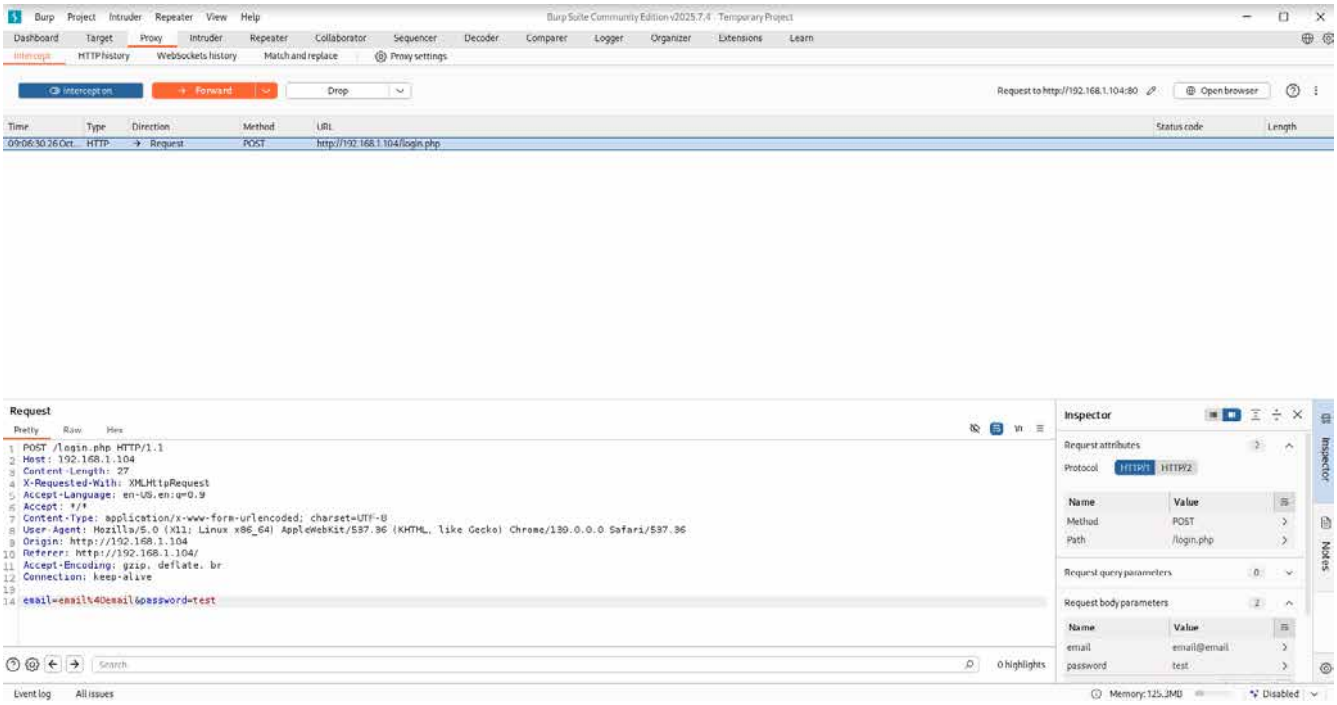


Рис. 3.2. Досліджуваний запит, який відправляє додаток до веб-серверу

Проведемо атаку SQLi в ручному режимі з використанням оператора UNION, щоб отримати корисну інформацію, а саме чи існує вразливість обробки SQL-запитів у веб-додатку. Даний процес наведено на рисунку 3.3.

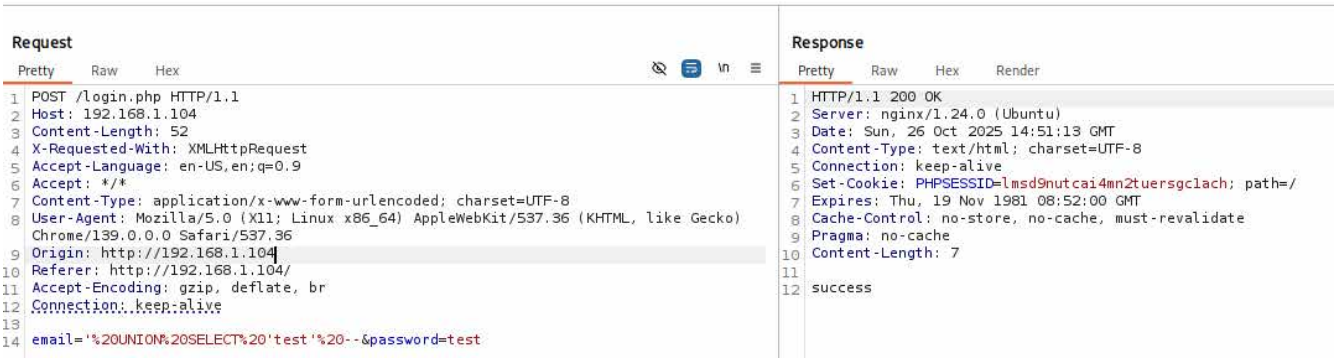


Рис. 3.3. Дослідження можливості експлуатації вразливості

Отже, отримавши підтвердження можливості експлуатації вразливості, для подальшого дослідження буде використано інструмент SQLMap для автоматизованого сканування на вразливість SQLi методом перебору. Для цього можна сформувати текстовий документ, в якому зберегти приклад POST HTTP запиту, який був попередньо отриманий:

- sqlmap -r "/home/kali/Desktop/request.txt" -p email --batch –dbs;

або вказати посилання з шляхом файлу login.php, а також вказати значення, які передаються до бази даних:

- sqlmap -u "http://192.168.1.104/login.php" --data="email=INJECT&password=test" -p email --batch –dbs.

При виконанні запиту, ми отримаємо дані, які зображені на рисунку 3.3:

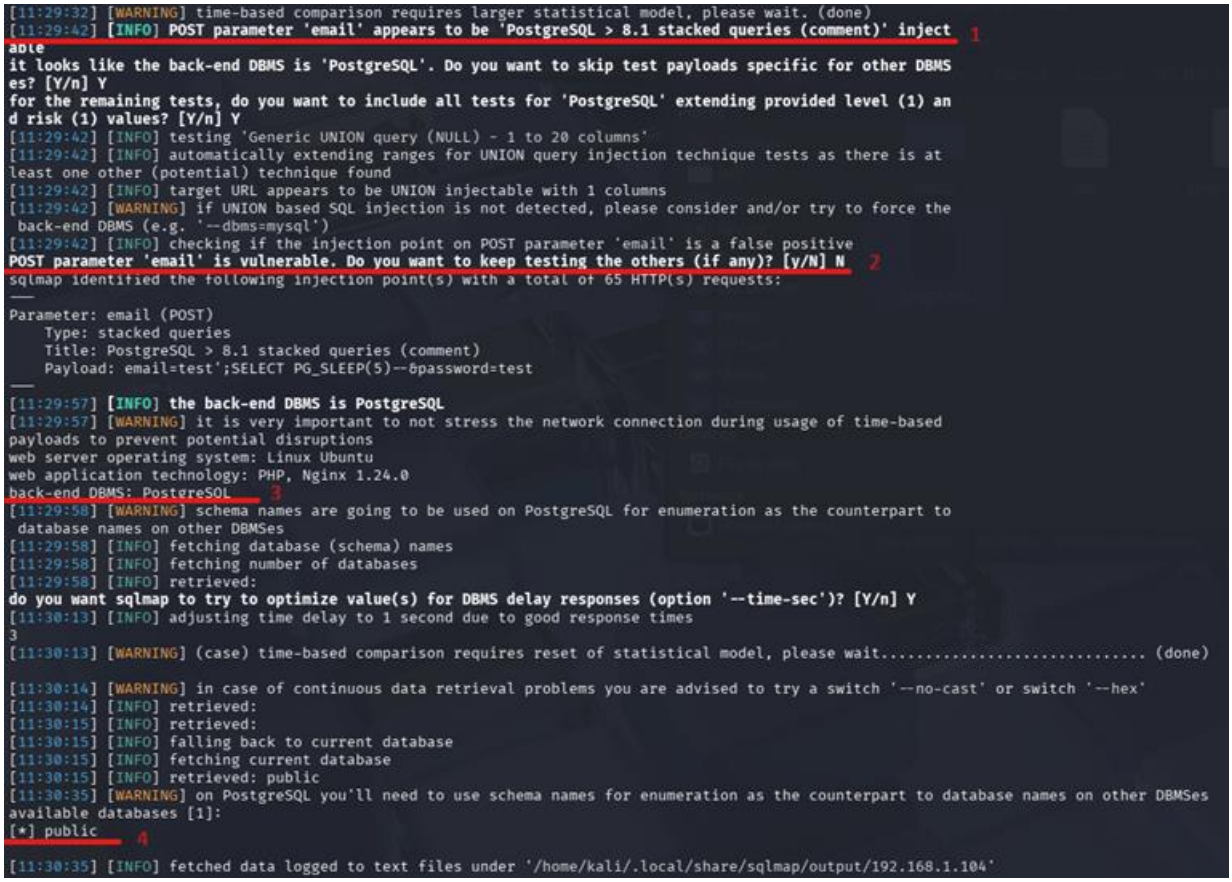


Рис. 3.3. Отримані дані при скануванні SQLMap

де основну інформацію було виділено:

1. повідомлення про наявну вразливість SQLi у параметрі email у атаці типу stacked queries;
2. запит на продовдження та експлуатацію SQLi;
3. виявлення назви продукту БД;
4. виявлення назви існуючої БД, а саме «public».

Отримавши назву бази даних, ми можемо отримати назву таблиць:

- sqlmap -r "/home/kali/Desktop/request.txt" -p email --batch -D public -tables;

назву колонок:

- sqlmap -r "/home/kali/Desktop/request.txt" -p email --batch -D public -T users –columns;

дамп пам’яті таблиці:

- qlmap -r "/home/kali/Desktop/request.txt" -p email --batch -D public -T users –dump.

Всі отримані записи, SQLMap зберігає у log-файлі на хості, як показано на рисунку 3.4:

```
web server operating system: Linux Ubuntu
web application technology: PHP, Nginx 1.24.0
back-end DBMS: PostgreSQL
Database: public
Table: users
[4 columns]
+---------------+---------+
| Column        | Type    |
+---------------+---------+
| email         | varchar |
| id            | int4    |
| username      | varchar |
| userpassword  | varchar |
+---------------+---------+

sqlmap resumed the following injection point(s) from stored session:
---
Parameter: email (POST)
    Type: stacked queries
    Title: PostgreSQL > 8.1 stacked queries (comment)
    Payload: email=test';SELECT PG_SLEEP(5)--&password=test
---
web server operating system: Linux Ubuntu
web application technology: PHP, Nginx 1.24.0
back-end DBMS: PostgreSQL
Database: public
Table: users
[2 entries]
+----+--------------------+----------+--------------+
| id | email              | username | userpassword |
+----+--------------------+----------+--------------+
| 1  | andriy@example.com | andriy   | hash_here    |
| 2  | df@df              | df       | df           |
+----+--------------------+----------+--------------+
```

Рис. 3.4. Виявленні дані БД в ході експлуатації вразливості

Паралельно проведенню дослідження експлуатації вразливості веб-додатку, на кінцевій точці був увімкнений агент XDR-системи Wazuh, який в ході моніторингу журналів подій веб-серверу Nginx, а саме access.log виявив веб-атаку, як показано на рисунку 3.5, а також зазначено приклад розгорнутого алерту, щодо успішності проведеної атаки, на рисунку 3.6:

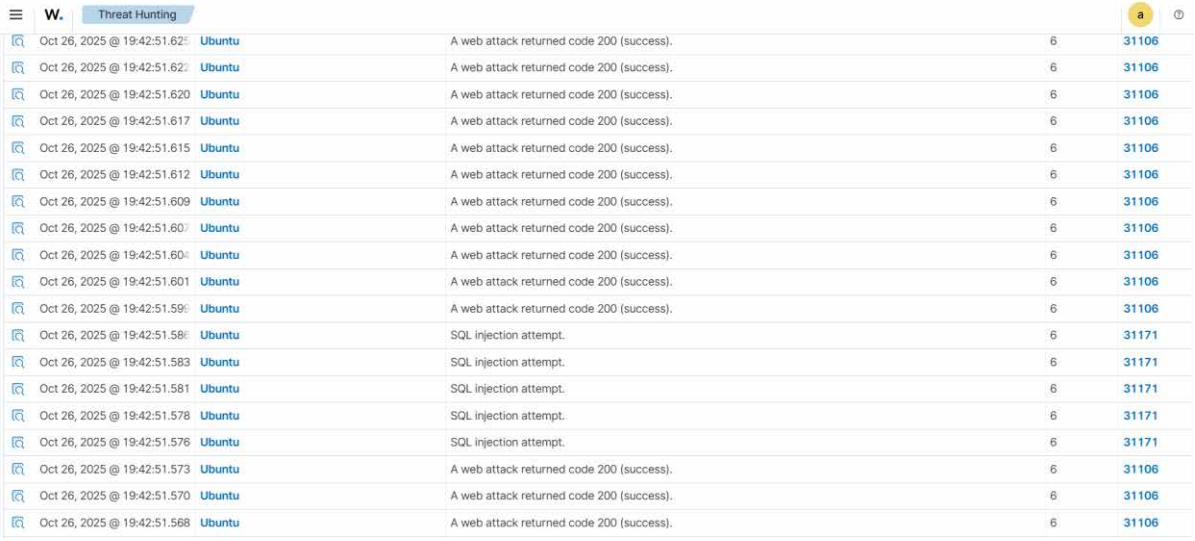


Рис. 3.5. Спрацювання алертів у системі Wazuh

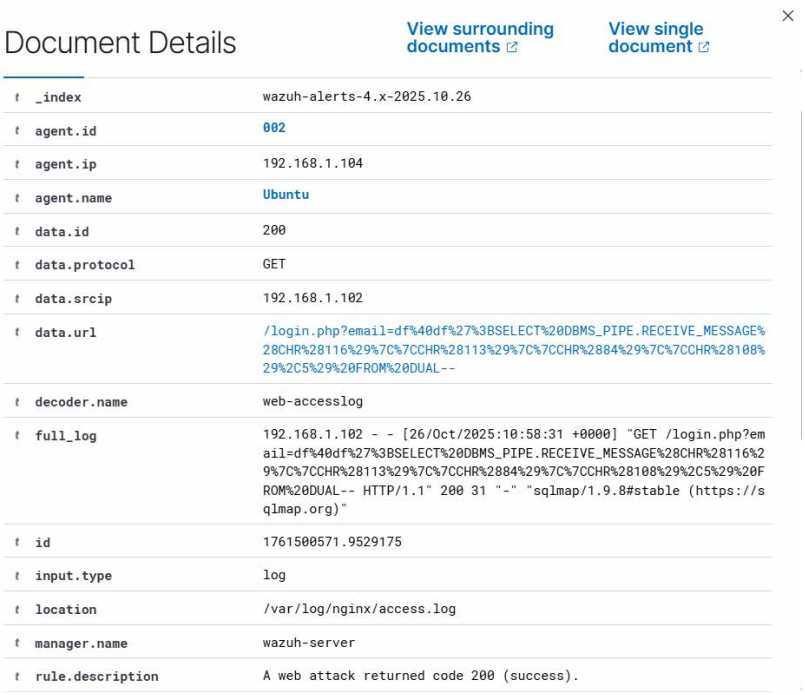


Рис. 3.6. Розгорнутий алерт спрацювання на успішно проведену атаку

3.2 Аналіз вразливості додатку та мінімізація впливу загрози

Підрозділ призначений опису вразливого фрагменту коду, що містить технічні пояснення та оцінку ризиків, а також реалізації безпечного коду, який унеможливить використання SQL-ін’єкції.

Початковий фрагмент вихідного коду має наступний формат:

```
$sql = "SELECT userpassword FROM users WHERE email = '$email'";
$result = pg_query($conn, $sql);
if ($result && pg_num_rows($result) > 0) {
    $row = pg_fetch_assoc($result);
    if ($password === $row['userpassword']) {
        ***
    } *** }
```

Даний код є вразливий в наступних параметрах:

- використовується рядкова конкатинація параметру $email без параметризації;
- конструкція без параметризації дозволяє зловмиснику впроваджувати SQL-ін’єкцію у значення $email, що дозволяє змінити логіку виконуваного запиту.

Для усунення вразливості SQL-рядку, запити було відокремлено від SQL-команди, де сервер обробляє значення параметру як символьний рядок, а не частину SQL-синтаксису. Це здійснюється за допомогою оператора pg_query_params, де підстановочним маркером запиту є $1 (див. Додаток А.2), а параметр $email передається як масив аргументів:

```
$sql = "SELECT userpassword FROM users WHERE email=$1";

$result = pg_query_params($conn, $sql, array($email));
```

Використання оператору pg_query_params є найпростішим та найефективнішим методом усунення вразливості SQLi, що не вимагає великих змін у коді, проте дає надійний захист від типових атак. Код файлів PHP представлений у Додатках А.2 та А.3.

3.3 Впровадження заходів безпеки для захисту середовища

З метою захисту веб-середовища, було впроваджено брандмауер ModSecurity 3.0 з офіційного репозиторію OWASP ModSecurity онлайн-платформи GitHub [23]. В описі цього підрозділу зазначено алгоритм дій з впровадження та налаштування захисту. Для початку роботи слід завантажити актуальну версію сервісу Nginx, що використвується веб-сервером, у тимчасову папку з назвою «rebuild» за допомогою вбудованої команди Linux «wget»:

```
wget https://nginx.org/download/nginx-1.26.2.tar.gz -O /tmp/rebuild/nginx-1.26.2.tar.gz

tar -zxvf /tmp/rebuild/nginx-1.26.2.tar.gz -C /tmp/rebuild
```

Наступний крок у створенні модуля захисту – це копіювання репозиторію конектора ModSecurity з платформи GitHub та розпакування вивантаженого архіву [24]:

```
wget https://github.com/owasp-modsecurity/ModSecurity-nginx/releases/download/v1.0.3/modsecurity-nginx-v1.0.3.tar.gz -O /tmp/rebuild/modsecurity-nginx-v1.0.3.tar.gz

tar -zxvf /tmp/rebuild/modsecurity-nginx-v1.0.3.tar.gz -C /tmp/rebuild/
```

В завантаженій директорії є вбудовані файли для автоматичної конфігурації модуля безпеки, де за об’єкт захисту було взято тимчасово створений Nginx:

```
cd /tmp/rebuild/nginx-1.26.2

./configure --with-compat --add-dynamic-module=/tmp/rebuild/modsecurity-nginx-v1.0.3

make

sudo make install
```

В результаті, ми отримали необхідний модуль: «ngx_http_modsecurity_module.so».

Його необхідно добавити до теки веб-сервісу, що здійснюється базовими командами Linux:

```
sudo cp objs/ngx_http_modsecurity_module.so /etc/nginx/modsec/

sudo ln -sf /etc/nginx/modules-available/ngx_http_modsecurity_module.so /etc/nginx/modules-enabled/ngx_http_modsecurity_module.so
```

Для запуску даного модуля середовищем серверу, необхідно внести зміни в перший блок конфігураційного файлу сервісу Nginx:

```
load_module modules/ngx_http_modsecurity_module.so;
```

Останнім кроком для налаштування роботи системи захисту є встановлення необхідних спеціальних правил детектування, що будуть виявляти шкідливу або аномальну активність в мережі, і найкращим вибором серед існуючих інструментів є правила OWASP, створені некомерційною спільнотою, мета яких – підтримка та захист безпеки веб-застосунків. Даний набір правил є відкритим для всіх користувачів Інтернету, та доступний для завантаження у середовище Linux наступною командою [25]:

wget https://github.com/SpiderLabs/owasp-modsecurity-crs/archive/v3.0.0.tar.gz

Далі, в директорії «/etc/nginx/modsec/main.conf», що була створена для налаштування роботи модуля захисту, слід добавити наступні інструкції:

OWASP CRS v3 rules

Include /usr/local/owasp-modsecurity-crs-3.0.0/crs-setup.conf

Include /usr/local/owasp-modsecurity-crs-3.0.0/rules/*.conf

Завершуючим етапом буде налаштування файлу конфігурацій актуальних сайтів веб-серверу, вказавши наступні аргументи:

modsecurity on;

modsecurity_rules_file /etc/nginx/modsec/main.conf;

Для порівняльної оцінки захисту веб-застосунку під час веб-атаки з використанням програмного забезпечення SQLMap, проводиться остаточна перевірка функціонування модулю, вбудувавши у пошуковий запит корисне навантаження:

sqlmap -u "http://192.168.1.104/login.php" --data="email=INJECT&password=test" -p email --batch –dbs.

Як бачимо на рисунку 3.7, працездатність брандмауера підтверджена відповідю веб-сервера з помилкою 403:

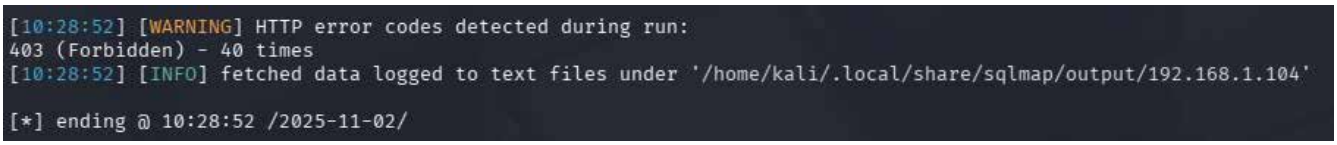


Рис. 3.7. Результат тестування захисту брандмауера ModSecurity

Також, оглянувши запис журналу помилок Nginx, ми бачимо причину блокування, а також сам шкідливий запит. На рисунку 3.8 можна ознайомитися із даною подією, в якості підтвердження працездатності модулю:

Рис. 3.8. Запис журналу помилок веб-серверу Nginx

В наступному підрозділі буде проведено дослідження ефективності захисту та покриття області детектування досліджуємого об’єкту.

3.4 Аналіз захисту веб-додатку з оцінкою ефективності

Оцінка ефективності відбудеться в декілька етапів, а саме методом аналізу подій до та після встановлення брандмауера веб-застосунку за допомогою матриці помилок (confusion matrix) [26], після чого можна здійснити порівняння отриманих метрик за допомогою формул. Нижче, наведено пояснення кожного елементу матриці, що досліджується [27]:

N (total) – загальна кількість запитів, що враховує запити з шкідливим навантаженням та легітимні входи користувача.

TP (true positive) – кількість успішно та правильно виявлених атак.

FP (false positive) – кількість помилкових спрацювань на події, що не є небезпечними.

FN (false negative) – кількість не виявлених подій (запитів) під час атаки.

TN (true negative) – кількість правильно визначених подій, де вхід користувача не є атакою.

Вхідні дані до впровадження ModSecurity отримані з трьох основних метрик, де загальна кількість запитів визначається добутком 40 шкідливих запитів, що були згенеровані SQLMap, 10 легітимних запитів, які імітували легітимні входи користувача та отримані дані зі спрацювань правил Wazuh, як показано на рисунку 3.9:

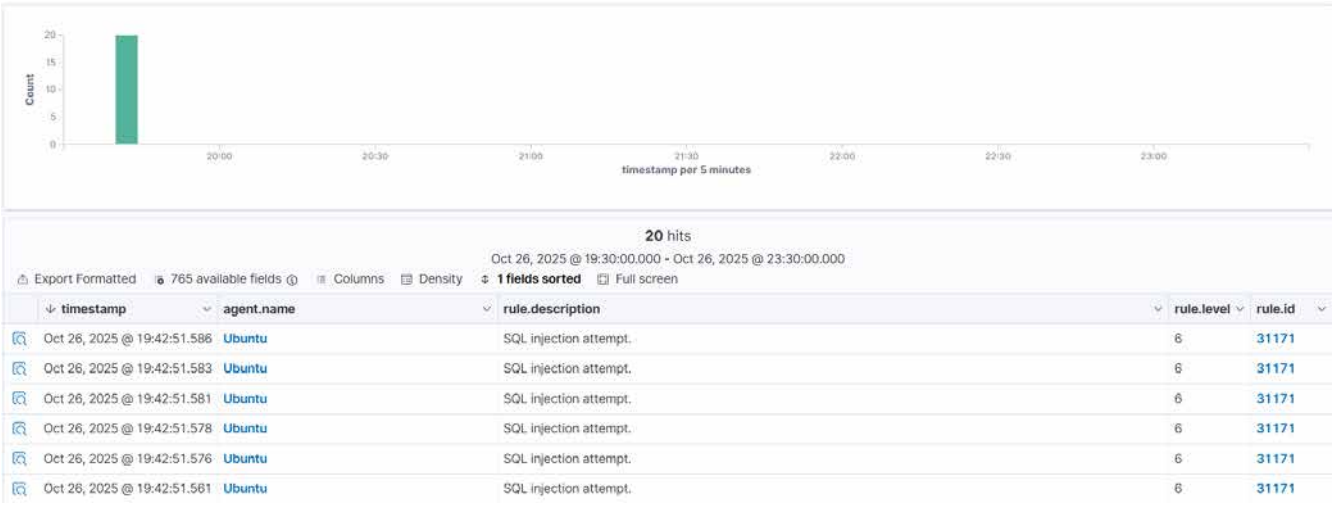


Рис. 3.9. Кількість спрацювань правил детектування Wazuh на загальну кількість запитів до впровадження WAF

$N_1 = 40 + 10 = 50$;

$TP_1 = 20$;

$FP_1 = 0$;

$FN_1 = 20$;

$TN_1 = 10$;

Вхідні дані після встановлення ModSecurity отримані аналогічним чином як в попередньому прикладі, де визначається добуток всіх запитів до СУБД з веб-додатку та кількість спрацювань правил детектування Wazuh, що зазначено на рисунку 3.10:

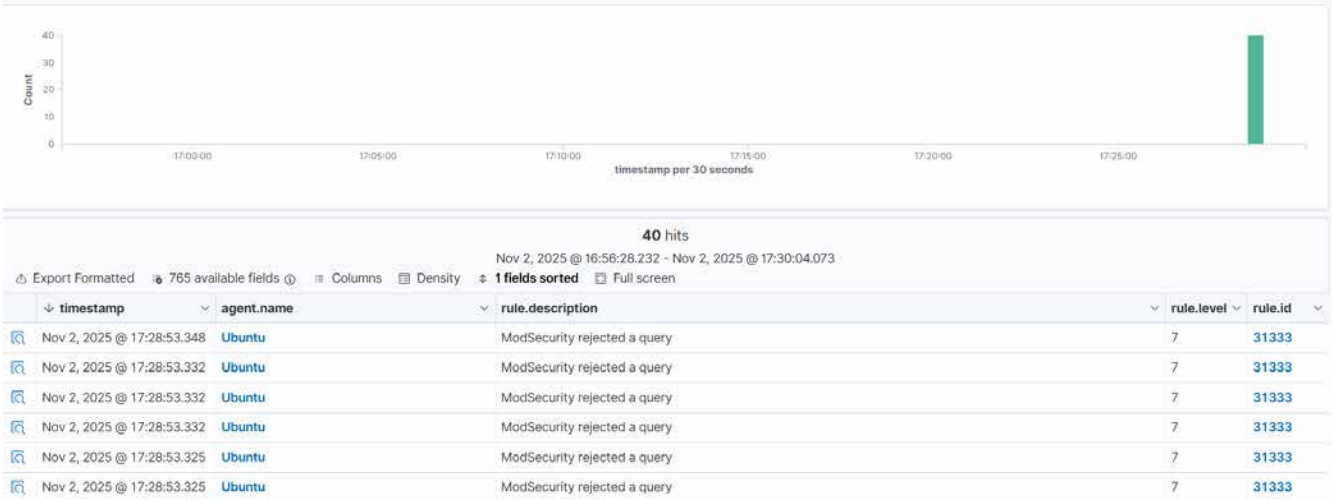


Рис. 3.10. Кількість спрацювань правил детектування Wazuh на загальну кількість запитів після впровадження WAF

$N_2 = 40$ (шкідливих запитів) + 10 (легітимних входів) = 50;

$TP_2 = 40$;

$FP_2 = 0$;

$FN_2 = 0$;

$TN_2 = 10$;

Далі, з отриманими даними, досліджуються метрики, для визначення ступеню ефективності до та після впровадження системи WAF [28]:

Precision – ступінь відповідності результатів вимірювання справжньому

значенню, що визначається за формулою: $Precision = \frac{TP}{TP+FP}$. (3.1)

$\text{Precision}_1 = \frac{20}{20+0} = 100\%$;

$\text{Precision}_2 = \frac{40}{40+0} = 100\%$;

Recall (TPR) – коефіцієнт істинно позитивних результатів, що визначається за формулою: $Recall = \frac{TP}{TP+FN}$. (3.2)

$\text{Recall}_1 = \frac{20}{20+20} = 50\%$;

$\text{Recall}_2 = \frac{40}{40+0} = 100\%$;

Accuracy – точність правильно класифікованих подій. Дана метрика має наступну формулу: $Accuracy = \frac{TP+TN}{N\ (total)}$. (3.2)

$\text{Accuracy}_1 = \frac{20+10}{50} = 60\%$;

$\text{Accuracy}_2 = \frac{40+10}{50} = 100\%$;

F1 – метрика середнього значення Precision (коефіцієнт влучності) та Recall (коефіцієнт детектування) $F1 = \frac{2 * Precision * Recall}{Precision+Recall}$. (3.3)

$\text{F1}_1 = \frac{2 * 1 * 0.5}{1 + 0.5} = 66.67\%$;

$F1_2 = \frac{2 * 1 * 1}{1+1} = 100\%$;

TNR – коефіцієнт специфічності, що означає кількість негативних результатів, які були правильно ідентифіковані, тобто кількість легітимних входів, на які система не відреагувала, та має формулу $TNR = \frac{TN}{TN + FP}$. (3.4)

$TNR_1 = \frac{10}{10+ 0} = 100\%$;

$TNR_2 = \frac{10}{10 + 0} = 100\%$;

Balanced Accuracy – вимір середньої чутливості до атак (Recall TPR) і чутливості до легітимних подій (TNR), що означає баланс при нерівних подій атаки і не атаки [29]. Визначається за формулою $Balanced\ Accuracy = \frac{TPR + TNR}{2}$. (3.5)

$Balanced\ Accuracy_1 = \frac{0.5 + 1}{2} = 0.75$;

$Balanced\ Accuracy_2 = \frac{1 + 1}{2} = 1$;

Youden’s J або індекс Юдена – оцінка ефективності тесту та визначає продуктивність діагностики за формулою: $Youden's\ J = TPR + TNR - 1$ [30]. (3.6)

$Youden's\ J_1 = 0.5 + 1 - 1 = 0.5$;

$Youden's\ J_2 = 1 + 1 - 1 = 1$;

Matthews Correlation Coefficient (MCC) – показник кореляції між рішенням системи та реальними подіями. Дана метрика являється найбільш точної для бінарної класифікації з наступною формулою:

$$MCC = \frac{\mathrm{T}P * TN - FP * FN}{\sqrt{(TP + FP) * (TP + FN) * (TN + FP) * (TN + FN)}}. \quad (3.7)$$

$$\mathrm{MCC}_1 = \frac{20 * 10 - 0 * 20}{\sqrt{(20) * (40) * (10) * (30)}} \approx 0.4083;$$

$$\mathrm{MCC}_2 = \frac{40 * 10 - 0 * 0}{\sqrt{(10) * (40) * (10) * (10)}} = 1;$$

У таблиці 3.1 можна переглянути зведені значення кожної з отриманих метрик оцінки ефективності, захисту та детектування:

Таблиця 3.1

Таблиця порівнянь ефективності

Метрика	без WAF	з WAF
Точність визначення події атакою (Precision)	100%	100%
Коефіцієнт детектування (Recall / TPR)	50%	100%
Точність класифікації всіх подій (Accuracy)	60%	100%
Інтегральна метрика узгодженості (F1)	66.67%	100%
Кількість правильно визначених легітимних подій	100%	100%
Збалансована точність між атакою та легітимною подією (Balanced Accuracy)	0.75	1
Оцінка ефективності тестування (Youdens J)	0.5	1
Кореляція детектування з реальними подіями (MCC)	0.4083	1

Остаточною метрикою дослідження буде двовибірковий z-тест пропорцій [31], що підтвердить справжність різниці між вимірюваннями, а не випадковість отриманих даних. Дана метрика потребує декілька етапів обчислень. В першу чергу визначаються змінні:

p_1 – частка виявлених атак до впровадження ModSecurity:

$$p_1 = \frac{TP}{TP + FN} = \frac{20}{20 + 20} = 0.5; \quad (3.8)$$

p_2 – частка виявлених атак після впровадження ModSecurity:

$$p_2 = \frac{TP}{TP + FN} = \frac{40}{40 + 0} = 1; \quad (3.9)$$

n_1 та n_2 – розмір вибірок, що означають кількість атак, та дорівнюють значенню 40.

Загальна формула z-тесту має наступний вигляд: $z = \frac{p_1 - p_2}{SE}$, (3.10)

де SE – це стандратна помилка для двох пропорцій:

SE = $\sqrt{p * (1 - p) * (\frac{1}{n_1} + \frac{1}{n_2})}$. (3.11)

Значенням p в даній формулі виступає пуліована пропорція (pooled proportion), та

вираховується формулою p = $\frac{TP_1 + TP_2}{n_1 + n_2}$. (3.12)

Визначивши формули для вирахування заданої метрики, можна приступати до обчислення:

p = $\frac{20 + 40}{40 + 40} = 0.75$;

SE = $\sqrt{0.75 * (1 - 0.75) * \left(\frac{1}{40} + \frac{1}{40}\right)} = \sqrt{0.009375} \approx 0.096824583$;

z = $\frac{1 - 0.5}{0.096824583} = 5.165$.

Отриманий результат можна інтерпретувати за допомогою сталого показника 1.96, тобто якщо |z| > 1.96 – довіра до вимірок більш 95%. Так як 5.165 > 1.96, можна стверджувати, що отримані результати дослідження не є випадковими. Це дозволяє зробити впевнений висновок у роботі, зокрема у ефективності WAF ModSecurity, як надійного засобу для забезпечення захисту веб-додатку.

ВИСНОВОК

Під час дослідження веб-атаки з використанням SQL-ін’єкції, було встановлено завдання для виконання, проаналізовано та досліджено методи детектування, захисту, а також інструменти для дослідження вразливості запитів до СУБД PostgreSQL.

Для створення лабораторних умов дослідження було розгорнуто серверну інфраструктуру, розроблено веб-застосунок та налагоджено взаємодію користувача з базою даних через вікно входу та реєстрації. В якості системи детектування було досліджено XDR Wazuh, а також програмне рішення брандмауера веб-застосунку – ModSecurity.

Кінцевим результатом даної роботи є аналіз вразливості, як зловмисник може нею скористатися, методи розвідки та ексфільтрації даних. Зокрема, математичними розрахунками було обчислено ефективність впровадження WAF, що збільшує область детектування, блокує шкідливі запити та унеможливлює доступ до більшості вразливостей коду. Згідно висновку проведеного аналізу, можна встановити, що використання систем детектування та захисту веб-трафіку є необхідним елементом для будь-якої сучасної системи, оскільки загроза витоку даних та порушення безпеки інформації є суттєвою та може призвести до серйозних наслідків, включно з повною компрометацією даних та зупинкою бізнес-процесів організації.

Дослідження показало, що навіть при використані стандартних методів валідації даних на стороні застосунку, інструменти як sqlmap можуть автоматизовано та детерміновано підбирати шкідливі запити, здійснювати розвідку інфраструктури. Ця робота є показовою та підтверджує необхідність багаторівневого підходу до захисту даних, так званий метод «defense in depth», для зменшення площини атаки та ризиків викрадення даних.

ДЖЕРЕЛА

1. OWASP Top Ten 2025. Top 10 Web Application Security Risks. URL: https://owasp.org/www-project-top-ten/ (дата звернення: 05.10.2025).
2. Sayenko, G., & Grinenko, T. (2019). Protecting web applications from SQL injections. “GLOBAL CYBER SECURITY FORUM 2019”. URL: https://openarchive.nure.ua/server/api/core/bitstreams/4bc35e97-c6a3-4155-bc68-70591fe4fd27/content (дата звернення: 12.10.2025).
3. Типи SQL-ін’єкцій (SQLi). Блог 2024. URL: https://corewin.ua/blog/types-of-sql-injection/ (дата звернення: 12.10.2025).
4. Тестування безпеки. SQLmap. SQL-ін’єкції та захоплення серверів баз даних. URL: https://training.qatestlab.com/blog/technical-articles/security-testing-sqlmap/ (дата звернення: 12.10.2025).
5. Федоренко А.А.,Осадчий Б. І., Коржик В. В. АНАЛІЗ МЕТОДІВ ВИЯВЛЕННЯ ВРАЗЛИВОСТЕЙ WEB-РЕСУРСІВ ДО SQL-ІН’ЄКЦІЙ. URL: https://journals.dut.edu.ua/index.php/dataprotect/article/view/2792/2691 (дата звернення: 13.10.2025).
6. SQL ін’єкції та захист від них. URL: https://foxminded.ua/sql-iniektsii/ (дата звернення: 14.10.2025).
7. Тестування вразливостей веб-додатків за допомогою Burp Suite. URL: https://hackyourmom.com/servisy/soft/burp-suite-owasp-top-10-owasp-juice-shop/ (дата звернення: 14.10.2025).
8. Уроки SQL. SQL-ін’єкції. URL: https://acode.com.ua/sql-injection/ (дата звернення: 14.10.2025).
9. Web application firewall (WAF): що це і як працює для захисту від атак хакерів. URL: https://netwave.ua/blog/web-application-firewall-waf-sho-ce/ (дата звернення: 14.10.2025).
10. IBM Verify Identity Access. Firewall Overview. URL: https://www.ibm.com/docs/en/sva/11.0.0?topic=firewall-overview (дата звернення: 14.10.2025).
11. EDR vs XDR vs SIEM: Який інструмент підійде саме вам? URL: https://www.softwareone.com/uk-ua/blog/articles/2024/07/23/edr-vs-exd-vs-siem-what-tool-is-right-for-you (дата звернення: 10.10.2025).

12. SIEM проти XDR: можливості та ключові відмінності.
URL: https://stellarcyber.ai/uk/learn/siem-vs-xdr/ (дата звернення: 10.10.2025).
13. Огляд платформи Wazuh.
URL: https://wazuh.com/ (дата звернення: 11.10.2025).
14. Nginx 1.24.0 Changes.
URL: https://nginx.org/en/CHANGES-1.24 (дата звернення: 14.10.2025).
15. Ubuntu Server. AppArmor Module.
URL: https://documentation.ubuntu.com/server/how-to/security/apparmor/ (дата звернення: 14.10.2025).
16. Ubuntu Server. Firewall (UFW).
URL: https://documentation.ubuntu.com/server/how-to/security/firewalls/ (дата звернення: 14.10.2025).
17. Getting started with Wazuh. Wazuh Architecture.
URL: https://documentation.wazuh.com/current/getting-started/architecture.html (дата звернення: 14.10.2025).
18. VMware virtual machine requirements and limitations.
URL: https://www.ibm.com/docs/en/storage-scale-ece/5.2.1?topic=infrastructure-vmware-virtual-machine-requirements-limitations (дата звернення: 15.10.2025).
19. Wazuh Installation. Virtual Machine.
URL: https://documentation.wazuh.com/current/deployment-options/virtual-machine/virtual-machine.html (дата звернення: 15.10.2025).
20. jQuery JavaScript Library. Nuget overview.
URL: https://www.nuget.org/packages/jquery/ (дата звернення: 15.10.2025).
21. Asynchronous JavaScript and XML technique.
ULR: https://developer.mozilla.org/en-US/docs/Glossary/AJAX (дата звернення: 15.10.2025).
22. Wazuh agent. Deploying Wazuh agents on Linux endpoints.
URL: https://documentation.wazuh.com/current/installation-guide/wazuh-agent/wazuh-agent-package-linux.html (дата звернення: 15.10.2025).
23. OWASP ModSecurity. GitHub reference.
URL: https://github.com/owasp-modsecurity/ModSecurity (дата звернення: 16.10.2025).
24. ModSecurity Issues. ModSecurity 3.0 interferes with nginx even when disabled.
URL: https://github.com/owasp-modsecurity/ModSecurity/issues/3336 (дата звернення: 16.10.2025).

25. Habr article. Інструкція по встановленню динамічного модулю на веб-сервер Nginx.
URL: https://habr.com/ru/articles/437032/ (дата звернення: 16.10.2025).
26. IBM. What is a confusion matrix?
URL: https://www.ibm.com/think/topics/confusion-matrix (дата звернення 16.10.2025).
27. Godil A. Bostelman R. Shackleford W. Hong T. Shneier M.Performance Metrics for Evaluating Object and Human Detection and Tracking Systems. P. 5-10
URL: https://nvlpubs.nist.gov/nistpubs/ir/2014/NIST.IR.7972.pdf.
28. Google Machine Learning. Класифікація: точність, повнота, влучність і пов’язані метрики.
URL: https://developers.google.com/machine-learning/crash-course/classification/accuracy-precision-recall?hl=uk (дата звернення: 16.10.2025).
29. Blog: Balanced Accuracy: When Should You Use It?
URL: https://neptune.ai/blog/balanced-accuracy (дата звернення: 16.10.2025).
30. Wikipedia. Youden's J statistic.
URL: https://en.wikipedia.org/wiki/Youden%27s_J_statistic (дата звернення: 16.10.2025).
31. PennState: Introduction to Mathematical Statistics. Comparing Two Proportions.
URL: https://online.stat.psu.edu/stat415/lesson/9/9.4 (дата звернення: 16.10.2025).

ДОДАТОК

Додаток А.1. Лістинг коду файлу index.html

```
<!DOCTYPE html>
<html>
  <head lang="en">
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link rel="stylesheet" href="style.css">
    <title>My Web Site</title>
  </head>
  <body>
    <header>
      <h2 class="logo">Logo</h2>
      <nav class="navigation">
        <a href="#">Home</a> <button class="btnLogin-popup">Login</button>
      </nav>
    </header>
    <div class="main-container"> <h1>This is test page</h1> </div>
    <div class="wrapper">
      <span class="icon-close"><ion-icon name="close"></ion-icon></span>
      <div class="form-box login">
        <h2>Login</h2>
        <form class="login-form" action="login.php" method="POST">
          <div class="input-box">
            <span class="icon"><ion-icon name="mail"></ion-icon></span>
            <input type="email" name="email" required>
            <label>Email</label>
```

```
        </div>
        <div class="input-box">
            <span class="icon"><ion-icon name="lock-closed"></ion-icon></span>
            <input type="password" name="password" required>
            <label>Password</label>
        </div>
        <button type="submit" class="btn">Login</button>
        <div class="login-register">
            <p>Don`t have an account? <a href="#" class="register-link">Register</a> </p>
        </div>
    </form>
</div>

<div class="form-box register">
    <h2>Registration</h2>
    <form class="register-form" action="register.php" method="POST">
        <div class="input-box">
            <span class="icon"><ion-icon name="person"></ion-icon></span>
            <input type="text" name="username" required>
            <label>Username</label>
        </div>
        <div class="input-box">
            <span class="icon"><ion-icon name="mail"></ion-icon></span>
            <input type="email" name="email" required>
            <label>Email</label>
        </div>
        <div class="input-box">
```

```
            <span class="icon"><ion-icon name="lock-closed"></ion-icon></span>
            <input type="password" name="password" required>
            <label>Password</label>
          </div>
          <button type="submit" class="btn">Register</button>
          <div class="login-register">
            <p>Already have an account?
              <a href="#" class="login-link">Login</a>
            </p>
          </div>
        </form>
      </div>
    </div>
    <script src="jquery-3.7.1.min.js"></script>
    <script>
      $(document).ready(function() {
        $('.login-form').on('submit', function(event) {
          event.preventDefault();
          $.ajax({
            url: 'login.php',
            type: 'POST',
            data: $(this).serialize(),
            success: function(response) {
              if (response === 'success') { window.location.href = 'home.html'; }
              else { alert(response); }
            },
            error: function() { alert('Сталася помилка при вході.'); }
```

```
            });
        });
    });

    $(document).ready(function() {
        $('.register-form').on('submit', function(event) {
            event.preventDefault();
            $.ajax({
                url: 'register.php',
                type: 'POST',
                data: $(this).serialize(),
                success: function(response) {
                    alert('Реєстрація пройшла успішно!');
                    $('.register-form')[0].reset(); },
                error: function(xhr, status, error) { alert('Сталася помилка при реєстрації: ' + error); }
            });
        });
    });
</script>
<script src="script.js"></script>
<script type="module"
src="https://unpkg.com/ionicons@7.1.0/dist/ionicons/ionicons.esm.js"></script>
<script nomodule src="https://unpkg.com/ionicons@7.1.0/dist/ionicons/ionicons.js"></script>
</body>
</html>
```

Додаток А.2. Лістинг коду файлу login.php

```
<?php
session_start();
ob_start();
$host = "localhost";
$port = "5432";
$dbname = "postgres";
$user = "postgres";
$password = "nubip";
$conn = pg_connect("host=$host port=$port dbname=$dbname user=$user password=$password");
if (!$conn) { die("Connection failed: " . pg_last_error()); }
if ($_SERVER["REQUEST_METHOD"] == "POST") {
   $email = $_POST['email'];
   $password = trim($_POST['password']);
   $sql = "SELECT userpassword FROM users WHERE email=$1";
   $result = pg_query_params($conn, $sql, array($email));
   if ($result && pg_num_rows($result) > 0) {
      $row = pg_fetch_assoc($result);
      if ($password === $row['userpassword']) {
         $_SESSION['user_email'] = $email;
         echo "success";
         exit();
      } else { echo "Неправильний пароль."; }
   } else { echo "Неправильний email."; }
} pg_close($conn); ?>
```

Додаток А.3. Лістинг коду файлу register.php

```
<?php
$host = "localhost";
$port = "5432";
$dbname = "postgres";
$user = "postgres";
$password = "nubip";
$conn = pg_connect("host=$host port=$port dbname=$dbname user=$user password=$password");
if (!$conn) { die("Connection failed: " . pg_last_error()); }
if ($_SERVER["REQUEST_METHOD"] == "POST") {
   $username = $_POST['username'];
   $email = $_POST['email'];
   $userpassword = $_POST['password'];
   $result = pg_query_params($conn, "INSERT INTO users (username, email, userpassword) VALUES
($1, $2, $3) RETURNING id", array($username, $email, $userpassword));
   if ($result) {
      $row = pg_fetch_assoc($result);
      $last_id = $row['id'];
      echo "Дані успішно додані. ID: " . $last_id; }
      else { echo "Помилка: " . pg_last_error($conn); }
   }
pg_close($conn); ?>
```