Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Інформаційно-аналітичних технологій та менеджменту
(повна назва)
Кафедра Інформатики
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти другий (магістерський)

РОЗРОБКА СКАНЕРА ВИЯВЛЕННЯ ВРАЗЛИВОСТЕЙ ВЕБСАЙТА ТА ДОСЛІДЖЕННЯ МЕТОДІВ ЗАХИСТУ ВІД РІЗНИХ ТИПІВ АТАК
(тема)

Виконав:
студент 2 курсу, групи ІНФм-22-3
Лопатінський Андрій Анатолійович
(прізвище, ініціали)

Спеціальності 122 Комп'ютерні науки
(код і повна назва спеціальності)

Тип програми освітньо-професійна

Освітня програма Інформатика
(повна назва освітньої програми)

Керівник доц. Руденко Д.О.
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри _____ Кобилін О.А.
(підпис) (прізвище, ініціали)

2024 р.

Харківський національний університет радіоелектроніки

Факультет Інформаційно-аналітичних технологій та менеджменту
(повна назва)

Кафедра Інформатики
(повна назва)

Рівень вищої освіти другий (магістерський)

Спеціальність 122 Комп'ютерні науки
(код і повна назва)

Тип програми освітньо-професійна

Освітня програма Інформатика
(повна назва освітньої програми)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«____»_____ 2024 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові Лопатінському Андрію Анатолійовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Розробка сканера виявлення вразливостей вебсайту та дослідження методів захисту від різних типів атак

затверджена наказом по університету від 14 листопада 2023 року № 1351Ст

2. Термін подання студентом роботи до екзаменаційної комісії 02 січня 2024 р.

3. Вихідні дані до роботи теоретичні відомості про типи вразливостей вебдодатків, модулі Python, теоретичні відомості про техніки захисту вебдодатків, файли для навантаження для тестування вебсайту.

4. Перелік питань, що потрібно опрацювати в роботі
1. Огляд методів пентестингу вебсайту.
2. Огляд методів захисту вебсайту від типових вразливостей.
3. Формування файлів навантаження для тестування вебсайту.
4. Програмна реалізація сканування сайту на різні типи атак.
5. Аналіз методів захисту вебсайту.

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (п.5 включається до завдання за рішенням випускової кафедри) основні типи вебвразливостей, основні типи вебвразливостей, техніки та методи захисту вебсайту від атак, програмна реалізація сканера

6. Консультанти розділів роботи (п.6 включається до завдання за наявності консультантів згідно з наказом, зазначеним у п.1)

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1	Отримання завдання на кваліфікаційну роботу	03.11.2023	
2	Аналіз завдання, підбір літератури	10.11.23-12.11.23	
3	Аналіз літератури з досліджуваної проблеми	12.11.23-13.11.23	
4	Аналіз технічних засобів	14.11.23-17.11.23	
5	Розробка методів	18.11.23-22.11.23	
6	Програмна реалізація	23.11.23-27.11.23	
7	Оформлення пояснювальної записки	30.11.23-03.12.23	
8	Перевірка на плагіат	04.01.2024	
9	Рецензування	06.01.2024	
10	Підготовка презентації та доповіді	08.01.2024	
11	Занесення роботи в електронний архів	10.01.2024	
12	Попередній захист кваліфікаційної роботи	10.01.2024	

Дата видачі завдання 3 листопада 2023 р.

Студент _____
(підпис)

Керівник роботи _____ доц. Руденко Д.О.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ/ABSTRACT

Пояснювальна записка до кваліфікаційної роботи: 79 с., 1 табл., 25 рис., 40 джерел.

ВЕББЕЗПЕКА, ВРАЗЛИВОСТІ ВЕБСАЙТІВ, СКАНУВАННЯ ВРАЗЛИВОСТЕЙ, SQL INJECTION, CROSS-SITE SCRIPTING (XSS), CROSS-SITE REQUEST FORGERY (CSRF), СИСТЕМИ ВИЯВЛЕННЯ ВТОРГНЕНЬ (IDS), СИСТЕМИ ПОПЕРЕДЖЕННЯ ВТОРГНЕНЬ (IPS), ОЦІНКА РИЗИКІВ, УПРАВЛІННЯ БЕЗПЕКОЮ ВЕБСАЙТІВ

Об’єктом дослідження є вебдодаток з навмисно низьким рівнем захисту від різних типів атак.

Метою дослідження є тестування різних методів атак на вебсайт та дослідження методик захисту від них.

У результаті дослідження здійснена програмна реалізація сканування на вразливості до таких типів атак: SQL Injection, Cross-Site Scripting, Server-Side Template Injection, Remote Code Execution. Також реалізовані додаткові функції для перевірки безпеки вебсайту.

WEB SECURITY, WEBSITE VULNERABILITIES, VULNERABILITY SCANNING, SQL INJECTION, CROSS-SITE SCRIPTING (XSS), CROSS-SITE REQUEST FORGERY (CSRF), INTRUSION DETECTION SYSTEMS (IDS), INTRUSION PREVENTION SYSTEMS (IPS), RISK ASSESSMENT, WEB SECURITY MANAGEMENT

The object of the research is a web application with a deliberately low level of protection against various types of attacks.

is to test various methods of attacks on the website and research methods of protection against them.

As a result of the research, a software implementation of scanning for vulnerabilities to the following types of attacks was carried out: SQL Injection, Cross-Site Scripting, Server-Side Template Injection, Remote Code Execution. Additional functions for checking website security have also been implemented.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

SQLi – SQL Injection (SQL ін’єкція)

XSS – Cross-Site Scripting (міжсайтовий запуск сценарію)

DOM – Document Object Model

SSTi – Server-Side Template Injection (впровадження шаблону на стороні сервера)

RCE – Remote Code Execution (віддалене виконання коду)

CSRF – Cross-Site Request Forgery (підробка міжсайтового запиту)

IDS – Instruction Detection Systems (системи виявлення інструкцій)

IDS – Instruction Prevention Systems (системи запобігання інструкцій)

ВСТУП

У сучасному світі, де інформаційні технології проникають у кожен аспект нашого життя, питання веббезпеки набуває особливої актуальності. Інтернет не лише значно спрощує наше повсякденне існування, але й відкриває нові можливості для бізнесу, освіти, науки та розваг. Однак, разом з розвитком цифрових технологій зро стає й рівень кіберзагроз, що ставить перед нами нові виклики в контексті захисту інформації та особистих даних.

Однією з ключових проблем в області веббезпеки є вразливості вебсайтів. Вразливості – це слабкі місця в коді вебсайту або в його конфігурації, які можуть бути використані зловмисниками для проведення атак. Ці атаки можуть мати різні форми: від несанкціонованого доступу до конфіденційних даних до повного паралізування роботи вебсайту.

Значення цієї проблематики зростає з кожним днем, оскільки все більше компаній та організацій використовують інтернет-ресурси для ведення своєї діяльності. Втрата даних, кібератаки або навіть короткочасні перерви в роботі можуть призвести до значних фінансових втрат, погіршення репутації, а іноді й до юридичних наслідків.

У цьому контексті важливим стає розуміння різних видів вебвразливостей, методів їх виявлення та принципів розробки ефективних заходів захисту. Вебвразливості можуть варіюватися від простих конфігураційних помилок до складних алгоритмічних недоліків. Найбільш поширеними типами вразливостей є SQL Injection, Cross-Site Scripting (XSS), Cross-Site Request Forgery (CSRF) і багато інших. Кожен з цих типів вимагає унікального підходу до виявлення та усунення.

Розробка сканерів вразливостей вебсайтів є одним з ключових напрямів у боротьбі з цифровими загрозами. Ці інструменти дозволяють автоматизувати процес виявлення потенційних слабких місць на вебсайтах, значно знижуючи ризики проведення успішних кібератак. Однак, розробка ефективного сканера

вимагає глибокого розуміння принципів функціонування вебдодатків, а також знання сучасних технік програмування та кібербезпеки.

У цій кваліфікаційній роботі проводиться дослідження різних аспектів розробки сканера вразливостей вебсайтів. Для цього необхідно проаналізувати існуючі методи виявлення вебвразливостей, вивчити підходи до захисту вебсайтів від різних типів атак та розглянути кроки розробки такого сканера, включаючи планування, виконання та тестування. Цей дослідницький підхід надасть комплексне розуміння проблеми та допоможе в розробці ефективних стратегій для захисту вебресурсів.

1 ОГЛЯД ВРАЗЛИВОСТЕЙ ВЕБСАЙТІВ І МЕТОДІВ ЇХ ВИЯВЛЕННЯ

1.1 Основні положення про веббезпеку та важливість виявлення вразливостей

У сучасному цифровому столітті веббезпека стала неодмінною частиною нашого онлайн існування. З розвитком інтернету та збільшенням його доступності вебресурси стали невід'ємною частиною особистого, соціального та корпоративного життя. Веббезпека охоплює захист вебсайтів, вебдодатків та онлайн сервісів від різноманітних загроз, що можуть піддавати ризику інформацію та діяльність користувачів.

1.1.1 Загальний огляд веббезпеки

Історія веббезпеки є історією неперервної еволюції. Від простих паролів та базових методів шифрування у ранніх днях інтернету веббезпека розвинулась до складних багаторівневих систем захисту, що включають фільтрацію трафіку, шифрування даних, аутентифікацію та авторизацію користувачів, та багато іншого [1]. Зростання інтернет-загроз, таких як віруси, трояни, фішинг, шпигунські програми та вебатаки, спонукало до розвитку передових технологій та підходів у галузі кібербезпеки.

Сучасна веббезпека стикається з безліччю викликів, включаючи захист від різноманітних атак, таких як DDoS (розподілені атаки «відмова у сервісі»), SQL-ін'єкції, XSS (Cross-Site Scripting), CSRF (Cross-Site Request Forgery) та багато інших. Зловмисники постійно розробляють нові методи та стратегії для обходу захисних механізмів, що змушує фахівців у галузі веббезпеки невпинно розвивати та оновлювати свої знання та інструменти.

Одним з ключових аспектів веббезпеки є захист персональних даних

користувачів. Втрата або несанкціонований доступ до цих даних може призвести до серйозних наслідків, включаючи фінансові збитки та втрату довіри клієнтів [2]. Важливість цього аспекту підкреслюється законодавством у різних країнах, таким як Загальний регламент про захист даних (GDPR) в Європейському Союзі.

Крім особистих даних важливим аспектом є також безпека корпоративних систем. Компанії використовують вебресурси для забезпечення різноманітних бізнес-процесів, від обробки замовлень до спілкування з клієнтами та управління внутрішніми ресурсами. Несанкціонований доступ до цих систем може призвести до втрати конфіденційної інформації, фінансових збитків, а також порушення нормативних та юридичних вимог.

Важливим інструментом у сфері веббезпеки є сканування вразливостей. Сканери вразливостей дозволяють виявляти потенційні слабкі місця в вебдодатках та інфраструктурі перед тим, як вони будуть використані зловмисниками. Це надає можливість вчасно вжити заходів для запобігання можливих атак.

В цілому, веббезпека є складним та багатогранним напрямком, який вимагає постійного вдосконалення знань, вмінь та інструментів. У наступних розділах буде поглиблено проаналізовано різні види вебвразливостей та методи їх виявлення, що є ключовим для розуміння та підвищення рівня веббезпеки в цифровому світі.

1.1.2 Історія та еволюція вебвразливостей

В історії веббезпеки важливе місце займають вебвразливості, які постійно еволюціонували разом з розвитком інтернет-технологій. У ранні дні Інтернету, коли він ще був у своєму зародку та обмежувався академічними та

військовими мережами, питання безпеки часто ігнорувалися. Безпека не вважалася пріоритетною, адже мережа використовувалася виключно довіреними особами. Однак з появою всесвітньої мережі та її доступністю для широкої публіки ситуація різко змінилася.

Початок 1990-х років позначив собою зростання популярності інтернету серед загальної публіки. Це також ознаменувало появу перших великих кібератак та виявлення вразливостей. Найперші вебвразливості були відносно простими та часто пов’язані з помилками в конфігурації серверів або недоліками в дизайні протоколів та додатків.

З часом, як розвивалися вебтехнології, вразливості ставали все складнішими та різноманітнішими. Програмісти та розробники почали використовувати складніші технології, такі як динамічний контент, інтерактивні форми та складні бази даних. Це відкрило шлях для нових типів атак, таких як SQL-ін’єкції, де зловмисники могли вставляти шкідливі SQL-запити через вебформи, щоб отримати доступ до баз даних.

На початку 2000-х років було зареєстровано значне зростання кількості вебатак, серед яких поширеними стали Cross-Site Scripting (XSS) та Cross-Site Request Forgery (CSRF). XSS дозволяла зловмисникам вставляти шкідливі скрипти на вебсторінки, які потім виконувалися у браузерах інших користувачів. CSRF змушував веббраузери користувачів виконувати небажані дії на сайтах, до яких вони були авторизовані.

У нинішні дні, з розвитком хмарних технологій та мобільного інтернету, вебвразливості продовжують еволюціонувати. На порядок денний виходять питання захисту персональних даних, конфіденційності, а також безпеки Інтернету речей. З’явилися нові види атак, такі як ransomware (вимагання викупу за розблокування даних) та атаки на цілісність даних.

Еволюція вебвразливостей свідчить про постійну гонку озброєнь між зловмисниками та фахівцями з веббезпеки. З одного боку, розробники додатків та систем безпеки намагаються випередити зловмисників, розробляючи більш надійні та безпечні системи. З іншого боку, зловмисники невпинно шукають нові слабкі місця та методи обходу захисних механізмів. Ця динаміка є ключовим елементом у розвитку веббезпеки, що вимагає від фахівців

постійного вдосконалення та оновлення їх знань та навичок.

Враховуючи швидкий розвиток технологій та постійне збільшення кількості вебресурсів, можна очікувати, що проблематика вебвразливостей буде залишатися актуальною і надалі. Постійний аналіз тенденцій, вивчення нових атак та розробка ефективних методів захисту є важливою частиною боротьби з кіберзлочинністю та забезпечення безпеки в інтернет-просторі.

1.1.3 Потреба в скануванні вразливостей

Розвиток вебтехнологій та зростаюча залежність від онлайн сервісів в сучасному світі сприяли збільшенню важливості сканування вразливостей як ключового елемента веббезпеки. Сканування вразливостей – це процес виявлення слабких місць та потенційних загроз в системах безпеки вебсайтів та вебдодатків. Цей процес не тільки допомагає ідентифікувати вразливі точки, але й сприяє розробці стратегій для їх усунення та запобігання можливих атак.

З огляду на невпинний розвиток та ускладнення кіберзагроз, систематичне сканування вразливостей є необхідністю для будь-якого вебсайту або вебдодатку [3]. Це дозволяє виявляти нові вразливості, які можуть виникнути в результаті змін у коді, оновленні програмного забезпечення чи через нові зовнішні загрози. Сканування допомагає організаціям бути на крок попереду зловмисників, забезпечуючи своєчасне виявлення та виправлення вразливостей.

Автоматизовані сканери вразливостей стали важливим інструментом у боротьбі з кіберзлочинністю. Ці інструменти сканують вебсайти на предмет відомих вразливостей, використовуючи різноманітні бази даних та евристичні методи. Вони дозволяють швидко та ефективно оцінювати великі обсяги коду, забезпечуючи розширене охоплення та глибокий аналіз потенційних ризиків.

Проте, використання автоматизованих сканерів також стикається з певними викликами та обмеженнями. Вони не завжди можуть виявити складні або специфічні для певної системи вразливості. Деякі види атак, наприклад, ті, що базуються на логіці додатків, можуть залишатися непоміченими. Тому,

поряд з автоматизованим скануванням, важливо також проводити ручні перевірки та аудити безпеки, які допомагають виявити та усунути ці вразливості.

Методи сканування вразливостей постійно розвиваються, намагаючись відповісти на виклики змін ландшафту кіберзагроз. Це включає вдо сконалення алгоритмів штучного інтелекту та машинного навчання для кращого виявлення складних вразливостей, а також інтеграцію сканерів з іншими інструментами безпеки, такими як системи виявлення вторгнень та моніторингу мережевого трафіку.

Сканування вразливостей є фундаментальним компонентом у побудові надійної системи веббезпеки. Воно допомагає організаціям вчасно виявляти та реагувати на потенційні загрози, мінімізуючи ризики втрати даних, фінансових збитків та репутаційних втрат. Розвиток та впровадження передових методів сканування в поєднанні з іншими заходами безпеки забезпечує комплексний захист від постійно мінливих кіберзагроз у динамічному цифровому середовищі.

1.2 Основні типи вебвразливостей

У цьому розділі детально розглянемо основні типи вебвразливостей. Ці вразливості є найпоширенішими та найнебезпечнішими загрозами для веббезпеки. Основні типи включають:

- SQL Injection;Cross-Site Scripting (XSS);
- Cross-Site Request Forgery (CSRF);
- Server-Side Template Injection;
- Remote Code Execution;
- вразливості безпеки сесії;
- небезпечні перенаправлення та перехоплення;
- некоректна обробка помилок;
- вразливості залежностей та компонентів.

На рисунку 1.1 продемонстровано розподіл за кількістю типових вразливостей вебсайтів за типом атаки.

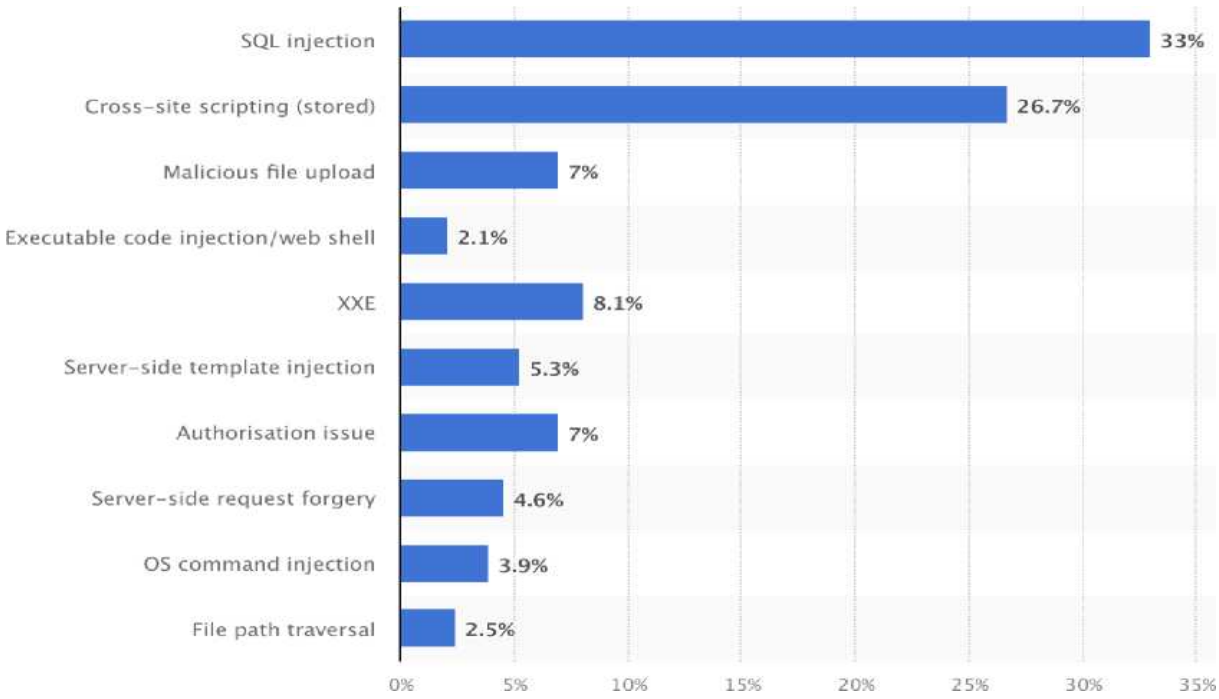

Рисунок 1.1 – Розподіл за кількістю типових вразливостей вебсайтів за типом атаки

1.2.1 SQL Injection

SQL Injection – це один із найстаріших і найнебезпечніших типів вебвразливостей. Він виникає, коли зловмисники вставляють або «ін'єктують» шкідливі SQL-команди в запити до бази даних через вебінтерфейс [4]. Це може статися через вебформи, параметри URL або навіть через вводи користувача, які передаються безпосередньо в SQL-запити (рис. 1.2).

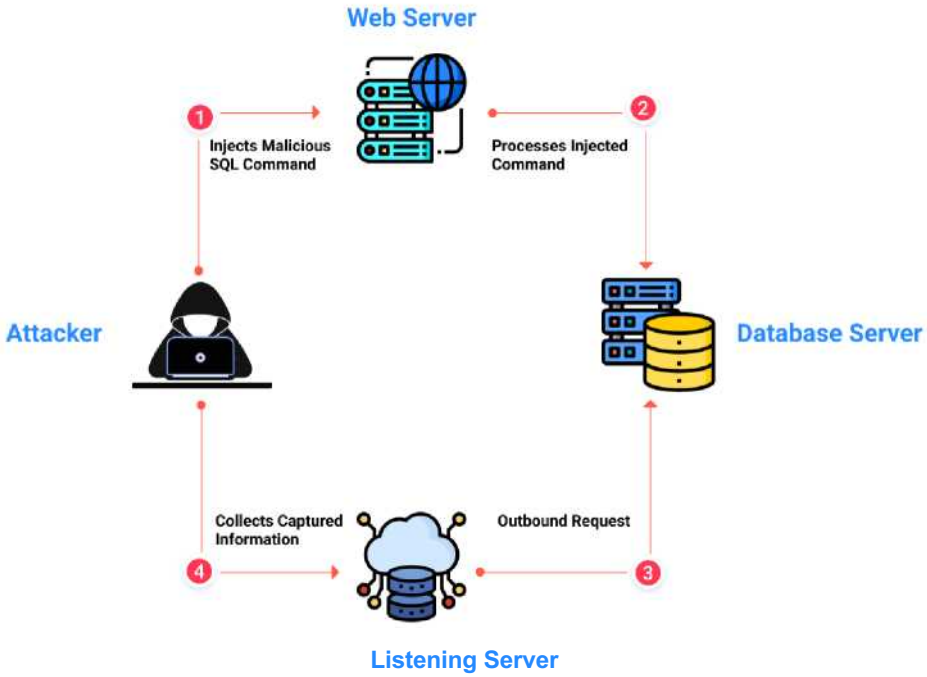

Рисунок 1.2 – Схема SQLi

SQL Injection є однією з найдавніших відомих вебвразливостей з першими згадками, що датуються кінцем 1990-х років. Вона швидко набула популярності серед зловмисників через свою ефективність та простоту використання [5]. З появою динамічних вебсайтів, що активно використовують бази даних, SQL Injection стала значною загрозою.

SQL Injection може бути класифікована на декілька типів (рис. 1.3):

– In-band SQLi, де атака та отримання результатів відбувається через один і той же канал (наприклад, через вебформу);

– Inferential SQLi, або «сліпий SQLi», де зловмисник не може безпосередньо побачити результат атаки, але може вивести інформацію, спостерігаючи за поведінкою вебдодатку;

– Out-of-band SQLi, що використовує різні канали для виконання атаки та отримання результатів, часто залежить від здатності сервера відсилати дані до зловмисника.

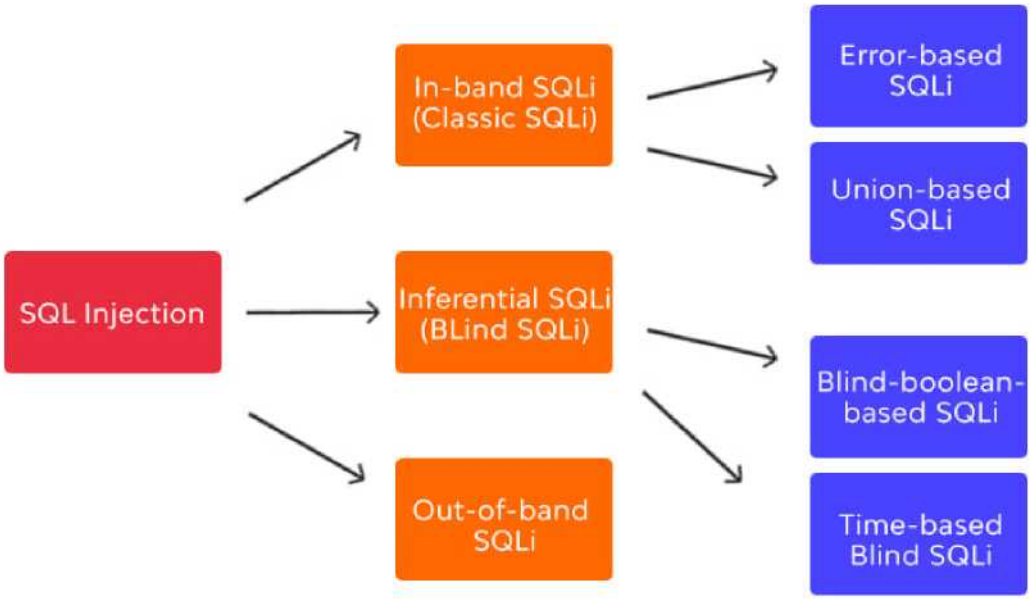

Рисунок 1.3 – Типи SQL injection

Наслідки SQL Injection можуть бути руйнівними. Вони включають викрадення конфіденційних даних, таких як особиста інформація користувачів, кредитні картки, корпоративні дані; знищення або зміна даних і навіть повне перехоплення контролю над базою даних. У найгірших випадках SQL Injection може призвести до компрометації цілої інфраструктури вебсайту.

Точки входу для SQL Injection часто знаходяться в місцях, де вебдодатки взаємодіють з базами даних без належних механізмів безпеки. Це можуть бути форми логіну [6], пошукові форми, URL-параметри, форми коментарів або будь-які інші місця, де вводяться дані користувачів.

Недостатня фільтрація та санітізація цих вводів є основною причиною виникнення вразливостей.

З роками SQL Injection залишається однією з найпоширеніших та найнебезпечніших вразливостей. Вона постійно займає високі позиції в рейтингах вебвразливостей, наприклад, у списках OWASP Top 10. Ця стабільність вказує на те, що багато вебдодатків досі вразливі до цього типу атак, незважаючи на зростання обізнаності про веббезпеку.

SQL Injection залишається важливою темою для дослідження та освіти у сфері веббезпеки, оскільки її наслідки можуть бути катастрофічними, а запобігання цьому типу атаки є ключовим для забезпечення безпеки онлайн-систем.

1.2.2 Cross-Site Scripting (XSS)

Cross-Site Scripting (XSS) є однією з найпоширеніших вебвразливостей. Вона виникає, коли зловмисники вставляють шкідливі скрипти на вебсторінки, що потім виконуються в браузері іншого користувача. Це може відбуватися через коментарі, форми вводу даних або інші інтерактивні елементи вебсайту (рис. 1.4).

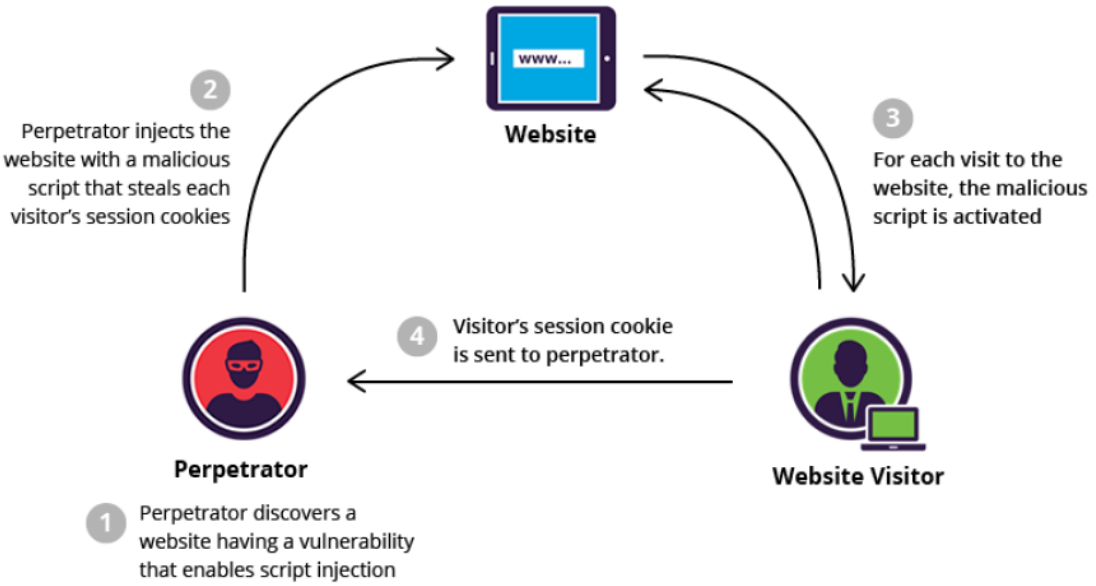

Рисунок 1.4 – Схема XSS

XSS вперше була ідентифікована у 1990-х роках і з тих пір стала однією з найбільш розповсюджених вебвразливостей. Із зростанням популярності вебдодатків, що активно використовують JavaScript та інші скриптові мови, XSS стала серйозною загрозою для веббезпеки.

XSS може бути класифікована на декілька основних типів (рис.1.5):

– Reflected XSS, де шкідливий скрипт відразу виконується при завантаженні сторінки, наприклад, через маніпулювання URL;

– Stored XSS, де шкідливий скрипт зберігається на сервері та виконується кожного разу, коли користувач завантажує певну сторінку;

– DOM-based XSS, що виникає внаслідок маніпуляцій з DOM (Document Object Model) вебсторінки без належної санітізації.

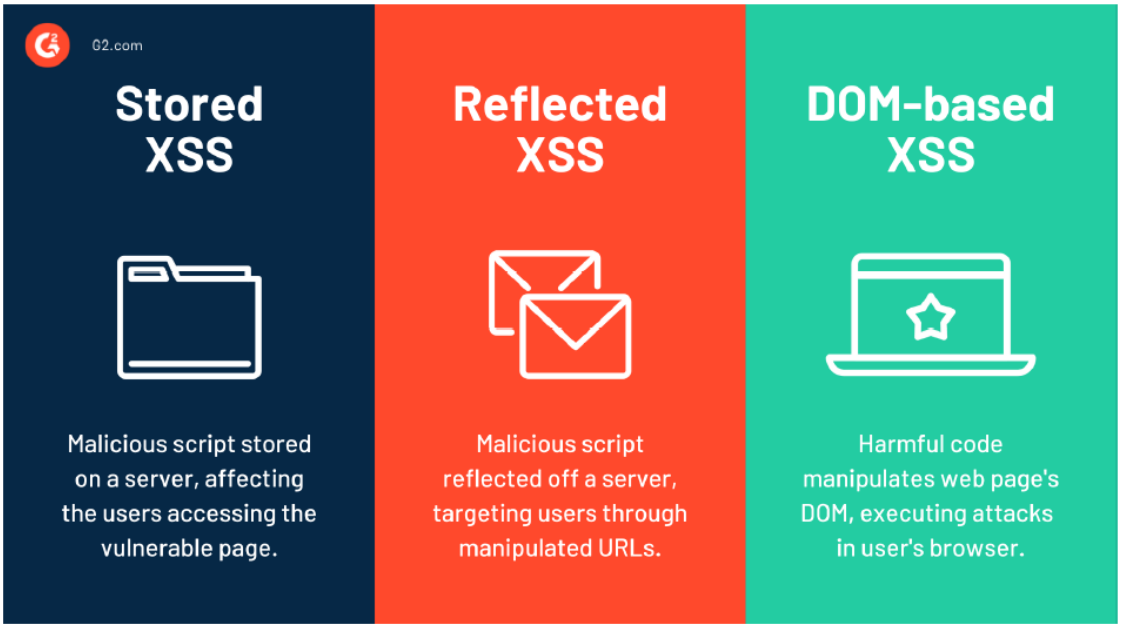

Рисунок 1.5 – Типи XSS

Наслідки атаки XSS можуть варіюватися від відносно невеликих, таких як маніпуляція з вмістом вебсторінки, до серйозних, включаючи крадіжку cookies [7], сесійних токенів, фінансової інформації, а також виконання шкідливих дій від імені жертви.

Точки входу для XSS часто знаходяться там, де вебдодатки приймають вводи користувача і повертають їх назад без належної обробки або санітізації. Це можуть бути форми для коментарів, пошукові поля, URL-параметри або будь-які інші місця, де користувач може ввести дані.

Згідно з дослідженнями, XSS залишається однією з найбільш часто виявлених вебвразливостей. Це підкреслює важливість розуміння цієї загрози та потреби в постійному моніторингу та оновленні захисних механізмів вебдодатків.

XSS представляє собою постійну загрозу в сучасному вебсвіті. Розуміння цієї вразливості, її різновидів та потенційних наслідків є важливим для забезпечення безпеки онлайн-систем і захисту конфіденційної інформації користувачів.

1.2.3 Cross-Site Request Forgery (CSRF)

Cross-Site Request Forgery (CSRF) – це вид вебатаки, що дозволяє зловмисникам змусити кінцевого користувача виконувати небажані дії на вебсайті, на якому цей користувач авторизований [8]. Через те, що запит відбувається від імені жертви, вебдодаток вважає його легітимним і виконує запит без додаткової перевірки (рис. 1.6).

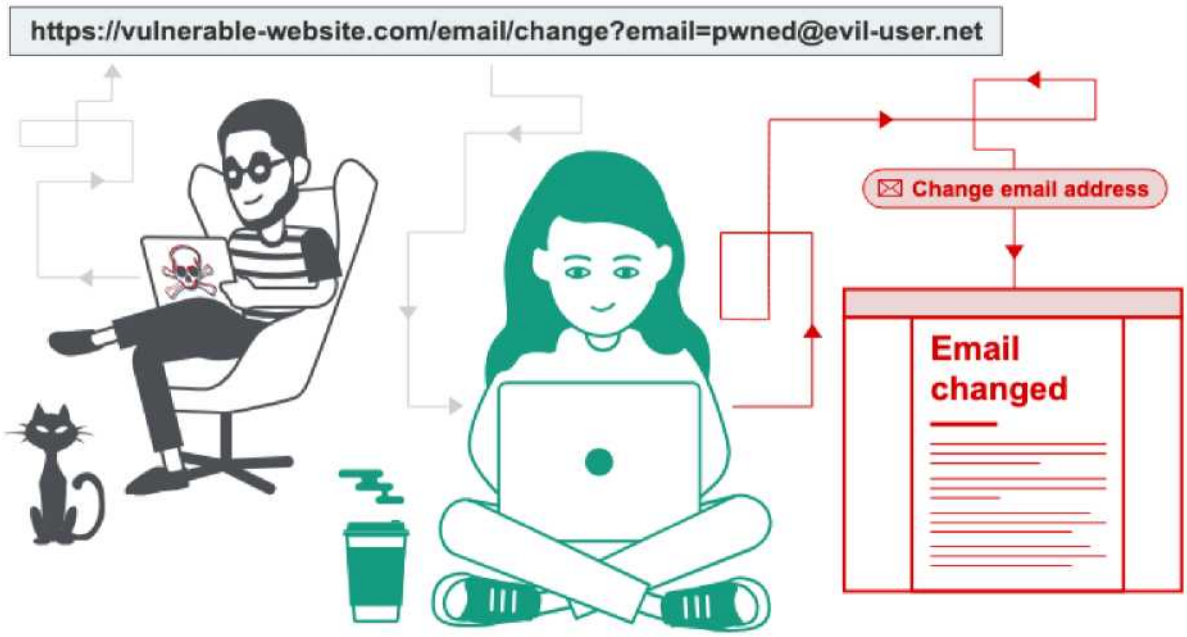

Рисунок 1.6 – Демонстрація CSRF

CSRF-атаки відомі з початку 2000-х років і вважаються одними з найстаріших типів вебатак. З тих пір вони стали однією з основних загроз для безпеки вебдодатків, особливо для тих, що мають значну кількість користувачів та обробляють конфіденційну інформацію.

Типи CSRF:

– традиційні CSRF-атаки, де зловмисник створює шкідливий запит і змушує жертву його виконати, часто використовуючи соціальну інженерію або маніпулюючи вебсайтами;

– автоматизовані CSRF-атаки, де використовуються скрипти або боти для генерації та відправлення запитів без відома користувачів.

Наслідки CSRF можуть варіюватися від зміни налаштувань користувача на вебсайті до здійснення несанкціонованих фінансових транзакцій або зміни

паролів. Враховуючи, що атака виконується в контексті сесії жертви, зловмисник може отримати контроль над її акаунтом.

Точки входу для CSRF часто знаходяться в місцях, де вебдодатки приймають важливі запити без додаткової перевірки автентичності. Це можуть бути будь-які дії, які виконуються через HTTP-запити, такі як зміна налаштувань користувача, відправлення повідомлень або транзакції.

Попри зростаючу обізнаність про безпеку, CSRF залишається серйозною загрозою для багатьох вебсайтів [9]. Ця стійкість вказує на складність вирішення цього типу вразливостей у вебдодатках, особливо в тих, де не використовуються сучасні методи захисту. На рисунку 1.7 можна побачити розподіл CSRF атак за сферою сайту.

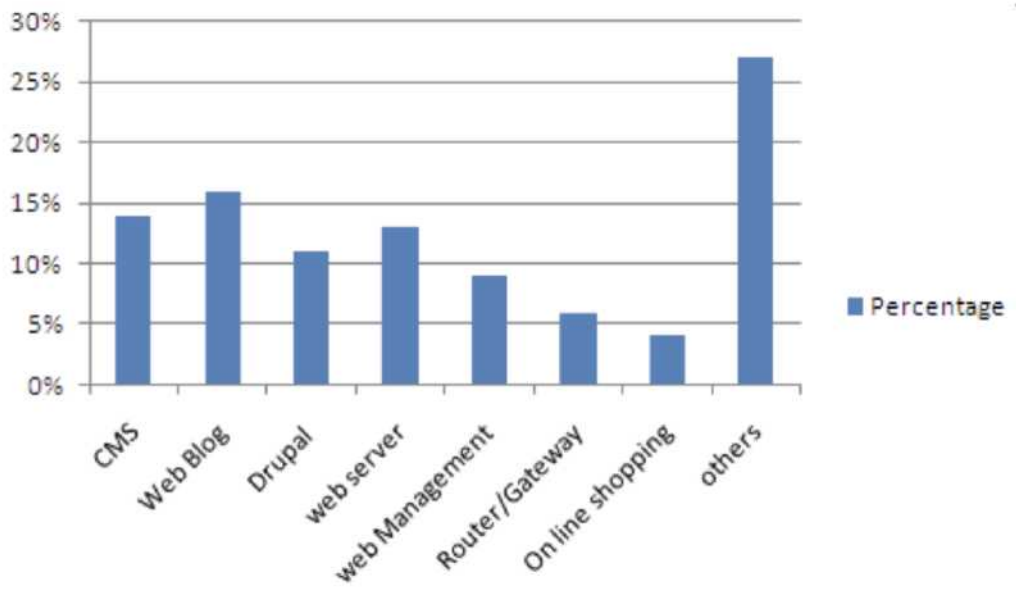

Рисунок 1.7 – Розподіл CSRF атак за сферою сайту

CSRF продовжує бути важливою темою у сфері веббезпеки, оскільки наслідки таких атак можуть мати серйозний вплив на користувачів та організації. Розуміння механізмів та потенційного впливу CSRF є ключовим для забезпечення безпеки в інтернет-середовищі.

1.2.4 Server-Side Template Injection

Server-Side Template Injection (SSTI) виникає, коли зловмисники можуть вводити або «ін'єктувати» шкідливий код у шаблони, які обробляються на стороні сервера. Ця вразливість виникає через неналежне санітізування або валідацію вводу, що дозволяє зловмисникам маніпулювати шаблонами та виконувати довільний код на сервері.

SSTI стали відомими у другій половині 2010-х років з розвитком вебдодатків, які активно використовують шаблони для генерації динамічного контенту. Розвиток фреймворків та шаблонізаторів сприяв зростанню функціональності, але також відкрив нові можливості для вразливостей.

Вразливості SSTI зазвичай виникають у вебдодатках, де користувач може впливати на структуру шаблону. Наприклад, якщо додаток використовує вводи користувача без належної обробки в шаблонах, зловмисник може вставити код, що буде виконаний на сервері.

SSTI можна класифікувати за типами в залежності від використовуваного шаблонізатора або мови програмування. Наприклад, вразливості можуть відрізнятися між шаблонізаторами, такими як Jinja2, Twig або FreeMarker [10].

Наслідки SSTI можуть бути серйозними, включаючи виконання довільного коду на сервері, що може призвести до витоку даних, компрометації серверів або розгортання шкідливого програмного забезпечення (ПЗ).

Точки входу для SSTI часто пов'язані з місцями, де додаток використовує вводи користувачів для генерації відповідей. Це може бути будь-який функціонал, який використовує вводи користувача в шаблонах, наприклад, форми зворотного зв'язку, динамічні URL-адреси або пошукові запити.

Хоча SSTI не так поширені, як інші типи вразливостей, вони представляють значний ризик через свій потенційний вплив. Атаки SSTI можуть призвести до серйозних порушень безпеки, оскільки вони дозволяють виконання коду на сервері.

Розуміння механізмів та потенційного впливу SSTI є важливим для

забезпечення безпеки сучасних вебдодатків, особливо тих, що активно використовують шаблони для генерації контенту.

1.2.5 Remote Code Execution

Remote Code Execution (RCE) – це вид вебвразливості, що дозволяє зловмисникам виконувати довільний код на віддаленому сервері або комп'ютері. Це може статися через вразливі додатки, некоректно налаштовані системи або через використання шкідливого вводу в програмах. RCE дозволяє атакувальнику отримати контроль над системою, виконувати команди, змінювати налаштування або використовувати систему для подальших атак.

RCE існує стільки ж, скільки і мережеві комп'ютери. З розвитком інтернету та комп'ютерних мереж RCE стала однією з найсерйозніших загроз, оскільки вона відкриває шлях до повного контролю над вразливими системами.

Механізм RCE часто включає використання багів або вразливостей у програмному забезпеченні. Наприклад, недоліки в обробці зовнішніх даних, таких як HTTP-запити або вводи користувачів, можуть дозволити атакувальнику вставляти шкідливий код [11]. Інші поширені точки входу включають некоректну обробку файлів, помилки у скриптах або неналежну валідацію вводу.

Типи RCE:

– Exploit-based RCE, де використовуються специфічні баги у програмному забезпеченні для виконання коду;

– Injection-based RCE, подібно до SQL Injection або XSS, де шкідливий код вводиться через стандартні механізми вводу, такі як форми або URL-параметри.

Наслідки RCE можуть бути руйнівними. Вони включають викрадення даних, встановлення шкідливого ПЗ, створення ботнетів, використання серверів для розповсюдження спаму або фішингу, а також серйозні системні порушення.

Точки входу для RCE часто пов’язані з вразливостями у програмному забезпеченні, які не були виявлені або не були виправлені. Це може включати застаріле ПЗ, неправильно налаштовані сервери, слабкі механізми аутентифікації або відсутність шифрування.

RCE залишається однією з найбільших загроз у сфері кібербезпеки. Із зростанням кількості підключених пристроїв та складності мережевих систем, потенціал для RCE-атак зростає.

Вплив RCE на сучасний цифровий світ є значним. Від корпоративних мереж до особистих пристроїв, ризик RCE-атаки є загрозою, яку не можна ігнорувати [12]. Розуміння механізмів та потенційного впливу RCE є ключовим для підвищення рівня безпеки в мережевому середовищі.

1.2.6 Вразливості безпеки сесії

Вразливості безпеки сесії – це серйозна проблема у веббезпеці, що стосується способів, якими вебдодатки управляють та зберігають стан користувача (або сесійну інформацію). Сесії є фундаментальним компонентом більшості вебдодатків, оскільки вони дозволяють системам «пам’ятати» користувачів між різними запитами. Проте, неналежне управління сесіями може призвести до ряду вразливостей, що ставлять під загрозу конфіденційність та безпеку користувачів.

Однією з основних проблем є неналежне управління сесійними токенами або cookies. Якщо сесійні токени легко передбачити або якщо вони передаються через незахищені канали, зловмисники можуть перехопити або відтворити ці токени, отримавши таким чином несанкціонований доступ до облікових записів користувачів.

Ще одна проблема пов’язана з тривалістю сесії. Дуже тривалі або постійні сесії збільшують ризик несанкціонованого доступу, особливо якщо користувач використовує публічні або спільні комп’ютери. Крім того, неправильне закінчення сесії після виходу користувача може залишити відкритими «двері» для потенційних атак.

Проблеми з безпекою сесій також виникають, коли вебдодатки неправильно використовують механізми сесій для управління даними користувача. Наприклад, зберігання надмірної або чутливої інформації в сесійних cookies без належного шифрування може призвести до витоку даних.

Вразливості безпеки сесії можуть мати серйозні наслідки, включаючи втрату конфіденційності користувачів, крадіжку облікових записів та інші форми кіберзлочинності [13]. Ці вразливості підкреслюють необхідність ретельного проектування та управління сесіями в рамках загальної стратегії веббезпеки.

1.2.7 Небезпечні перенаправлення та перехоплення

У контексті виявлення вразливостей вебсайтів, небезпечні перенаправлення та перехоплення є критичними аспектами безпеки, що вимагають особливої уваги. Ці вразливості можуть бути використані зловмисниками для маніпулювання користувачами, викрадення конфіденційної інформації або навіть введення користувачів в оману щодо намірів та ідентичності вебсайту.

Небезпечні перенаправлення відбуваються, коли вебсайт неправильно обробляє вхідні URL-адреси, дозволяючи атакувальникам перенаправляти користувачів на шкідливі сайти. Це може бути здійснено через неналежно санітізовані параметри запитів або через вразливості у скриптах обробки вводу.

Перехоплення здійснюється, коли атакувальник має можливість перехоплювати комунікацію між користувачем та вебсайтом. Це часто відбувається в незахищених або публічних мережах, де зловмисники можуть використовувати методи типу «man-in-the-middle» для перехоплення даних.

Вебсайти, які страждають від цих вразливостей, ризикують втратою довіри користувачів, можливими юридичними наслідками через витік даних та репутаційними втратами. Крім того, перехоплення даних може призвести до серйозних витоків конфіденційної інформації.

1.2.8 Некоректна обробка помилок

Некоректна обробка помилок в контексті виявлення вразливостей вебсайту – це значний аспект веббезпеки, який часто упускається з уваги. Ця вразливість виникає, коли вебдодаток або сервер відображає помилки, що містять чутливу інформацію про систему, яку можна використовувати для подальших атак.

Під час виявлення вразливостей, аналіз помилок, які відображаються користувачу, є ключовим для зрозуміння потенційних слабких місць вебсайту. Інформація, отримана з помилок, може включати деталі про структуру бази даних, використовувані технології, внутрішню логіку додатку та навіть конкретні шляхи файлової системи [14].

До інформації, що може бути витягнута з помилок, належать:

– версії серверного ПЗ (наприклад, Apache, Nginx, SQL сервери);

– специфічні повідомлення про помилки баз даних, що вказують на проблеми з SQL запитами;

– інформація про серверні конфігурації або шляхи до файлів.

Зловмисники можуть використовувати цю інформацію для планування специфічних атак, включаючи SQL Injection, шляхові траверсії файлової системи або навіть підготовки цілеспрямованих атак на певні версії ПЗ, які використовуються на сервері.

Для вебсайтів, особливо тих, що зберігають конфіденційні дані або виконують фінансові операції, некоректна обробка помилок є значним ризиком. Вона може вести до втрати даних, компрометації системи та навіть юридичних наслідків у випадку витоків конфіденційної інформації.

1.2.9 Вразливості залежностей та компонентів

Сучасні вебдодатки часто використовують велику кількість сторонніх компонентів для забезпечення широкого спектру функціональностей, від візуалізації даних до зв’язку з базами даних. Кожен з цих компонентів може

нести потенційні ризики безпеки, особливо якщо вони не регулярно оновлюються або не підтримуються їх розробниками [15].

Однією з основних проблем є застарілі компоненти, що часто містять відомі вразливості. Важливо, щоб розробники та адміністратори вебсайтів регулярно перевіряли та оновлювали всі використані бібліотеки та компоненти, щоб запобігти використанню вразливих версій.

Управління залежностями у великих вебдодатках може бути складним завданням через велику кількість використовуваних бібліотек та їх взаємозалежностей. Проте, це критично важливо для забезпечення безпеки та надійності вебдодатків.

1.3 Методи та техніки, які використовуються для експлуатації цих вразливостей

1.3.1 Експлуатація SQL Injection

Може бути реалізована через вебформи, URL-параметри або будь-які інші точки вводу даних, де вводи користувачів включаються в SQL запити.

За типами:

– In-band SQLi: зловмисник використовує ту саму комунікаційну лінію для введення шкідливого коду та отримання результатів. Найпростіший для використання, зазвичай включає ERROR-based SQLi (де помилки відображають корисну інформацію) та UNION-based SQLi (використання оператора UNION для об’єднання результатів запитів);

– Inferential SQLi (сліпий SQLi): не дає безпосереднього результату, але зловмисник може робити висновки про структуру бази даних на основі поведінки вебдодатку [16]. Це включає BOOLEAN-based SQLi (де відповідь змінюється залежно від того, чи є запит істинним чи хибним) та TIME-based SQLi (де відповідь затримується, підтверджуючи істинність запиту);

– Out-of-band SQLi: використовується, коли прямий відгук не можливий. Зловмисник використовує певні функції SQL для відправлення даних до сервера, який контролюється ним.

Найпростіший приклад – це введення ’ OR ’1’=’1 в поле логіну або паролю, що може привести до автоматичного входу без перевірки пароля (поширено в слабо захищених системах).

Зловмисник може використовувати SQL-команди для витягу даних з бази, наприклад, *UNION SELECT username, password FROM users,* щоб отримати список користувачів і паролів.

За допомогою SQL-ін’єкцій можна також видаляти таблиці, змінювати дані або навіть модифікувати структуру бази даних.

У більш складних випадках зловмисники можуть використовувати автоматизовані інструменти для виявлення та експлуатації SQL-ін’єкцій, а також використовувати складні запити для обходу фільтрів безпеки та отримання доступу до особливо захищених даних.

1.3.2 Експлуатація XSS

Експлуатація XSS, або Cross-Site Scripting включає в себе використання різноманітних технік для вставки шкідливого коду на вебсторінки, які потім виконуються в браузері жертви. Цей процес може бути реалізований у різних формах, від простих скриптів до більш складних атак, що залучають динамічні елементи вебсторінок.

Найпростішим прикладом XSS може бути введення шкідливого скрипту безпосередньо в текстове поле на вебсайті, таке як форма коментарів або пошукове поле [17]. Наприклад, користувач вводить *<script>alert(‘XSS’)</script>* у форму коментарів. Якщо вебсайт не санітізує цей ввід належним чином, цей JavaScript-код буде виконаний в браузері кожного, хто переглядає цей коментар, відображаючи спливаюче вікно.

Reflected XSS відбувається, коли шкідливий скрипт передається як частина запиту, наприклад, через URL. Зловмисник створює посилання з включеним у нього шкідливим скриптом і переконує жертву перейти за цим посиланням. При переході на такий URL, шкідливий скрипт виконується в браузері жертви. Наприклад, URL може виглядати так:

http://example.com/search?query=<script>alert('XSS')</script>. При відкритті такого посилання скрипт активується.

У випадку збереженого XSS, шкідливий код зберігається на сервері, наприклад, у базі даних, і виконується кожного разу, коли користувач взаємодіє з відповідним контентом. Наприклад, зловмисник може вставити шкідливий скрипт у свій профіль або в публікацію на форумі. Коли інші користувачі переглядають цей профіль або публікацію, шкідливий код автоматично виконується.

DOM-based XSS відбувається через маніпуляції з DOM вебсторінки. Це може включати зміни у URL, які потім обробляються JavaScript на стороні клієнта без належної фільтрації. Наприклад, URL *http://example.com/#<script>alert('XSS')</script>* може використовуватися для виконання скрипту безпосередньо через DOM.

1.3.3 Експлуатація CSRF

Експлуатація технік CSRF (Cross-Site Request Forgery) полягає у використанні того факту, що вебдодаток не перевіряє, чи дійсно користувач мав намір виконувати певну дію. Зловмисник створює шкідливий запит, який виконується в контексті сесії користувача. Це може включати перенаправлення або відправку запиту на сервер, де жертва вже аутентифікована.

Розширені приклади експлуатації CSRF:

– використання зображення для надсилання запиту: зловмисник може вставити зображення на форум або в соціальну мережу з URL-адресою, що вказує на виконання важливої дії на іншому сайті. Наприклад, *<img src= 'http://bank.com/transferMoney?amount=1000&to=attacker '>*. Якщо користувач вже аутентифікований на сайті банку, просте завантаження цього зображення може спричинити несанкціонований банківський переказ;

– використання CSRF в соціальній інженерії: зловмисник може використовувати елементи соціальної інженерії, наприклад, відправивши електронний лист із посиланням, що містить шкідливий запит. Посилання

може вести на здається легітимну сторінку, але при кліку на нього користувач без своєї відомості виконує дію на третьому сайті. Наприклад, лист може містити запрошення переглянути новий коментар у блозі, але насправді це призведе до зміни налаштувань електронної пошти;

– автоматизація CSRF атак: зловмисники можуть використовувати скрипти для автоматизації та розсилки шкідливих CSRF-атак. Це може включати розробку форм, які автоматично відправляються, коли користувач завантажує шкідливу сторінку, виконуючи запити на зміну паролів або навіть видалення акаунтів;

– використання AJAX для надсилання запитів: за допомогою AJAX можна створювати більш складні CSRF-атаки, де запити відправляються асинхронно. Це дозволяє зловмисникам виконувати послідовні дії на сайті, імітуючи поведінку легітимного користувача. Наприклад, спочатку може бути виконано запит на отримання токену, а потім відправлено інший запит, що використовує цей токен для авторизації дії.

Ці приклади демонструють, як техніки CSRF можуть бути використані для виконання несанкціонованих дій від імені жертви, використовуючи її вже існуючу сесію. Це ставить під загрозу не тільки безпеку даних користувача, але й цілісність вебсайту.

1.3.4 Експлуатація RCE

Експлуатація технік Remote Code Execution (RCE) зазвичай включає в себе виявлення та використання вразливостей у програмному забезпеченні, що дозволяють зловмисникам виконувати довільний код на віддаленому сервері або комп’ютері. Ці техніки можуть бути різноманітними та залежать від конкретних уразливостей системи, її конфігурації або використовуваних технологій.

Один з цікавих прикладів експлуатації RCE –це використання вразливостей у вебсерверних скриптах або програмах. Наприклад, якщо вебдодаток використовує незахищену функцію для виконання команд системи

на основі вводу користувача, зловмисник може ввести спеціально сформований запит, що включає системні команди. Це може бути таким простим, як додавання ; rm -rf / до вводу в текстовому полі, яке призведе до видалення файлів на сервері, якщо цей ввід буде безпосередньо використано в команді оболонки.

Інший приклад – використання багатоступеневих RCE атак, де зловмисник спочатку використовує одну вразливість для отримання обмеженого доступу до системи, а потім використовує інші вразливості для ескалації привілеїв. Наприклад, спочатку може бути використана SQL Injection для отримання доступу до бази даних, а потім використовується незахищена функція системного журналування для виконання довільних команд.

Також RCE може бути реалізований через вразливості в бібліотеках і фреймворках, які використовуються вебдодатками. Наприклад, стара версія фреймворку може містити вразливість, яка дозволяє виконувати код через спеціально сформовані HTTP-запити.

Важливо відзначити, що RCE є однією з найнебезпечніших вебвразливостей, оскільки вона дозволяє зловмисникам виконувати будь-які дії в контексті сервера, включаючи викрадення даних, зміну або видалення інформації, а також розгортання шкідливих програм. Розуміння та виявлення таких вразливостей є ключовими для забезпечення безпеки вебдодатків.

1.3.5 Експлуатація інших типових вразливостей

Експлуатація вразливостей безпеки сесії часто включає в себе перехоплення або вгадування сесійних токенів. Наприклад, якщо вебдодаток відправляє сесійні cookies через незахищене з’єднання, зловмисник може використовувати атаку «man-in-the-middle» для перехоплення цих cookies. Отримавши ці дані, атакувальник може використовувати ці cookies для імітації сесії легітимного користувача, отримуючи доступ до його акаунту або конфіденційної інформації. Іншим прикладом може бути використання предиктивних алгоритмів для вгадування сесійних токенів на основі зразків

або зламу сесійних токенів через відомі слабкості в їх генерації.

У сценарії небезпечних перенаправлень зловмисники створюють шкідливі URL-адреси, які автоматично перенаправляють користувача на шкідливі сайти. Це може бути використано для фішингових атак або розповсюдження шкідливого програмного забезпечення. Наприклад, зловмисник може вставити URL, який виглядає легітимно, але насправді перенаправляв користувача на сайт, що викрадає облікові дані. У випадку перехоплень, атакувальники можуть використовувати незахищені Wi-Fi мережі для перехоплення даних, що передаються між користувачем та сервером, що може включати паролі, фінансову інформацію та інші конфіденційні дані.

При некоректній обробці помилок детальні повідомлення про помилки можуть відкрити інформацію про внутрішню структуру системи. Наприклад, повідомлення про помилку SQL може вказувати на конкретні таблиці або поля бази даних, які можна використовувати для подальших SQL Injection атак. Або повідомлення про помилку сервера може включати шляхи до файлів або інформацію про використовуване серверне ПЗ, що дозволяє атакувальникам краще підготуватися до атаки.

Зловмисники часто використовують вразливості в застарілих або неналежно налаштованих компонентах вебдодатків. Наприклад, використання застарілої версії бібліотеки JavaScript може мати відому вразливість, яку можна використати для виконання шкідливих скриптів або перехоплення даних [18]. Атакувальники також можуть використовувати вразливості в плагінах або розширеннях, які інтегровані у вебдодатки, для отримання доступу до сервера або його даних.

1.4 Аналіз існуючих сканерів вразливостей

Важливість сканерів вразливостей у сучасному кіберпросторі не може бути переоцінена. Зі зростанням кількості вебдодатків та складності кібератак потреба в надійних інструментах для виявлення слабких місць стає все більш

актуальною. Ці інструменти допомагають забезпечити захист від широкого спектру вебвразливостей, від загальновідомих до дуже специфічних та високотехнологічних.

Сучасні сканери вразливостей пропонують різноманітність функцій: від автоматизованого сканування та генерації звітів до інтерактивних інструментів для ручного тестування безпеки. Ця гнучкість дозволяє використовувати їх у різних сценаріях, від невеликих стартапів до великих корпорацій.

Однак, використання сканерів вразливостей також має свої виклики. Наприклад, інколи вони можуть генерувати помилкові позитиви, вказуючи на вразливості, які насправді не представляють ризику. Також вони можуть пропускати деякі складні або нові види атак.

Інтеграція сканерів вразливостей у процес розробки ПЗ є критично важливою. Раннє виявлення та виправлення вразливостей може значно знизити ризики та вартість усунення проблем у майбутньому. Це стає особливо актуальним в контексті DevOps та неперервної інтеграції.

1.4.1 OWASP ZAP

OWASP ZAP є одним з найвідоміших інструментів в області безпеки вебдодатків. Розроблений в рамках Open Web Application Security Project (OWASP), цей інструмент є безкоштовним та відкритим для спільноти. Основні особливості ZAP включають:

– автоматичне та ручне сканування: ZAP надає можливості для як автоматичного, так і глибокого ручного сканування вразливостей;

– проксі-сервер: ця функція дозволяє перехоплювати та аналізувати трафік між браузером та вебдодатком;

– підтримка скриптів: ZAP підтримує різні скрипти для розширення його функціональності та автоматизації певних завдань;

– активне та пасивне сканування: інструмент дозволяє виявляти вразливості без впливу на функціональність вебдодатків (пасивне сканування) та з активним втручанням для детального аналізу (активне сканування).

1.4.2 Nessus

Nessus є одним з провідних комерційних інструментів для сканування вразливостей. Він широко використовується для сканування мереж, систем та додатків [19]. Характеристики Nessus включають:

– широкий спектр перевірок безпеки: Nessus надає більше тисячі перевірок для виявлення вразливостей у мережах, ПЗ та вебдодатках;

– оновлення безпеки: регулярні оновлення забезпечують виявлення найновіших вразливостей;

– зручний інтерфейс користувача: Nessus відомий своїм інтуїтивно зрозумілим користувацьким інтерфейсом, що робить його доступним для фахівців різного рівня;

– підтримка різних платформ: Nessus доступний для багатьох операційних систем, включаючи Windows, Mac OS та Linux.

1.4.3 Burp Suite

Burp Suite –це комплексний набір інструментів для тестування безпеки вебдодатків. Він включає ряд функцій для глибокого аналізу та тестування:

– Proxy-сервер: центральна функція Burp Suite, що дозволяє перехоплювати та аналізувати трафік;

– Scanner: автоматизований сканер вразливостей, який може виявляти ряд загальних вразливостей;

– Intruder: інструмент для автоматизованого відправлення запитів, що допомагає в тестуванні різних параметрів та функціональності вебдодатків;

– Repeater та Sequencer: інструменти для детального аналізу запитів та відповідей, а також для тестування сесій.

1.4.3 Вибір сканера

При виборі сканера вразливостей важливо враховувати специфіку вебдодатку, ресурси та потреби організації. Для малих компаній або проектів із обмеженим бюджетом безкоштовні інструменти, такі як OWASP ZAP, можуть бути відмінним рішенням. Для більших корпорацій або складних додатків, що вимагають більш глибокого аналізу та налаштувань, комерційні рішення, такі як Nessus або Burp Suite, можуть бути більш підходящими [20].

Важливо зазначити, що хоча сканери вразливостей можуть автоматично ідентифікувати багато вразливостей, вони не замінюють потребу в ручному тестуванні та експертній оцінці. Комбінація автоматизованого сканування з глибоким аналізом забезпечує найкращий рівень безпеки.

1.5 Постановка задачі дослідження

Розробка простого сканера вразливостей для вебсайтів має важливе значення у сучасному контексті кібербезпеки. Такий інструмент дозволяє швидко виявляти основні вразливості, як-от SQL ін’єкції, Cross-Site Scripting (XSS) та інші, що є критично важливим для захисту вебресурсів від потенційних атак. Простий сканер стає особливо корисним для малих та середніх підприємств, яким може не вистачати ресурсів для використання складних комерційних рішень. Такий інструмент також може служити як навчальний ресурс для початківців у сфері кібербезпеки, демонструючи їм основні принципи та методи виявлення вразливостей. Крім того, такий сканер може бути вихідною точкою для подальшої розробки та налаштування, щоб він відповідав специфічним потребам організації, забезпечуючи гнучкість та адаптивність у захисті вебсайтів.

Об’єктом дослідження є вебдодаток з навмисно низьким рівнем захисту від різних типів атак.

Метою дослідження є тестування різних методів атак на вебсайт та дослідження методик захисту від них.

Для досягнення мети необхідно вирішити такі завдання:

– створення набору алгоритмів, які ефективно виявляють SQL Injection, Cross-Site Scripting, Server-Side Template Injection та Remote Code Execution та Open Redirection, Clickjacking, тестування API безпеки;

– інтеграція таких бібліотек як requests, selenium, beautifulsoup4 для обробки вебзапитів, взаємодії з вебсторінками та аналізу вмісту;

– впровадження системи журналювання для відстеження діяльності сканера та результатів сканування;

– розробка інтуїтивно зрозумілого користувацького інтерфейсу, який дозволяє легко керувати процесом сканування та переглядати звіти;

– проведення тестування сканера.

2 МЕТОДИ ТА ПІДХОДИ ДО ЗАХИСТУ ВЕБСАЙТІВ ВІД АТАК

2.1 Огляд методів захисту вебсайтів

2.1.1 Загальний огляд стратегій безпеки вебсайтів

Безпека вебсайтів базується на трьох основних принципах: конфіденційності, цілісності та доступності. Конфіденційність забезпечує, що інформація залишається приватною та доступною тільки для уповноважених осіб. Цілісність забезпечує захист даних від несанкціонованих змін. Доступність означає, що ресурси вебсайту постійно доступні для легітимних користувачів.

Захист вебсайту можна розглядати на декількох рівнях, включаючи мережевий рівень (наприклад, використання брандмауерів та систем виявлення вторгнень), рівень застосунків (захист вебдодатків, наприклад, через валідацію вводу та санітізацію), та рівень даних (захист баз даних та систем зберігання даних).

Одним з ключових аспектів безпеки вебсайту є використання HTTPS, яке забезпечує шифрування даних, переданих між клієнтом та сервером. Це особливо важливо для захисту конфіденційної інформації, такої як паролі та платіжні дані, від перехоплення [21].

Постійне оновлення та застосування патчів до серверного ПЗ, вебдодатків та інших компонентів системи є критично важливим для запобігання вразливостям. Багато кібератак використовують відомі вразливості, які могли бути усунуті через своєчасне оновлення.

Керування доступом та розподіл привілеїв грає важливу роль у безпеці вебсайтів. Це включає в себе використання сильних паролів, двофакторної аутентифікації та принципу найменших привілеїв для обмеження доступу до критичних системних компонентів.

Ефективна стратегія безпеки також вимагає постійного моніторингу та швидкого реагування на безпекові інциденти. Це може включати використання інструментів моніторингу для відстеження підозрілої активності та розробку

плану реагування на інциденти для ефективного усунення виявлених проблем.

Регулярне резервне копіювання даних та здатність швидко відновити систему після інциденту є важливою частиною стратегії безпеки. Це забезпечує відновлення вебсайту у випадку атаки, яка призводить до втрати даних або пошкодження системи.

Важливим елементом захисту вебсайтів є санітізація та валідація вводу, особливо в тих місцях, де користувач може ввести дані. Це включає захист від SQL Injection, XSS та інших видів ін’єкцій. Валідація даних на стороні сервера гарантує, що лише вірні та безпечні дані обробляються додатком.

Розділення різних компонентів вебсайту (наприклад, фронтенд, бекенд, база даних) та їх розміщення в ізольованих мережевих сегментах може значно знизити ризик розповсюдження атаки у разі компрометації одного з компонентів. Це також сприяє кращому контролю доступу та обмеженню потенційного впливу вразливостей.

Захист від розподілених відмовних атак (DDoS) є ключовим для підтримки доступно сті вебсайтів. Це може включати використання спеціалізованих сервісів та інфраструктури для мінімізації впливу масових запитів на роботу вебсерверів.

Освітні програми та тренінги для розробників, адміністраторів та інших членів команди, які працюють з вебсайтом, є невід’ємною частиною стратегії безпеки. Освічений персонал краще здатний виявляти, запобігати та реагувати на потенційні загрози безпеки.

2.1.2 Історія та розвиток методів захисту від атак

В історії кібербезпеки, ранні дні Інтернету характеризувались відносною простотою та відкритістю мереж. Безпека часто спиралась на базові механізми аутентифікації та прості мережеві брандмауери [22]. Первинна увага приділялася фізичному захисту мережевої інфраструктури, а не програмному захисту даних.

Зі зростанням популярності Інтернету та вебдодатків зросла і кількість

загроз. Віруси, троянські програми, шпигунське ПЗ, і, особливо, атаки на вебдодатки, такі як SQL Injection, XSS, стали все більш поширеними. Це спричинило потребу в розвитку більш складних методів захисту.

З часом стало очевидним, що ефективний захист вимагає комплексного підходу. Це означало не лише удосконалення технологій брандмауерів та антивірусного захисту, але й розробку політик безпеки, процедур відповіді на інциденти, а також проведення регулярних аудитів та тренінгів.

З появою електронної комерції, онлайн-банкінгу та інших вебсервісів, безпека вебдодатків стала невід'ємною частиною захисту інформації. Розробники почали інтегрувати безпеку безпосередньо в процес розробки ПЗ, використовуючи методи, такі як безпечне програмування та автоматичне тестування вразливостей.

Шифрування стало стандартною практикою для захисту конфіденційних даних, передаваних через Інтернет. Протоколи, такі як SSL/TLS, стали основою для безпечної передачі даних, особливо в фінансових та електронних комерційних транзакціях.

З розвитком Інтернету з'явилася і проблема DDoS-атак. Це спонукало розробку спеціалізованих систем захисту, здатних мінімізувати вплив цих атак на доступність вебресурсів. Використання CDN (Content Delivery Networks) та спеціалізованого апаратного забезпечення стало поширеною практикою для захисту від цих загроз.

Освіта та навчання персоналу стали ключовими компонентами ефективної стратегії кібербезпеки. Зростання свідомості про загрози та впровадження найкращих практик серед розробників, адміністраторів та кінцевих користувачів вважаються одними з найефективніших способів запобігання інцидентів безпеки.

Поширення хмарних технологій та послуг внесло нові виклики та можливості для кібербезпеки. Хмарні провайдери почали пропонувати розширені інструменти та сервіси для захисту додатків та даних, розташованих у хмарі, включаючи автоматичні механізми виявлення та відповіді на загрози.

Таким чином, методи захисту від атак постійно еволюціонували, відповідаючи на зростаючу складність і різноманітність кіберзагроз. Ця

еволюція відображає не тільки технічний прогрес, але й зміну підходів до управління ризиками та культури безпеки в цифровому світі.

2.2 Техніки захисту та їх ефективність

2.2.1 Санітізація та валідація вводу

Санітізація та валідація вводу є ключовими аспектами безпеки вебсайтів. Ці практики допомагають попередити введення шкідливих даних та запобігти ін’єкційним атакам, таким як SQL Injection та XSS. Давайте розглянемо їх більш детально.

Санітізація вводу – це процес очищення введених даних від потенційно небезпечних символів або коду. Основна мета санітізації – забезпечити безпеку системи шляхом видалення або заміни потенційно небезпечного вводу. Наприклад, якщо користувач вводить текстовий рядок, який має включати тільки букви та цифри, то будь-які спеціальні символи, які можуть використовувати для ін’єкцій, будуть видалені.

Наприклад, користувач вводить наступний рядок в поле для коментарів: *<script>alert(‘Hacked!’);</script>*. Санітізація видаляє або замінює спеціальні символи та рядок стає безпечним: *alert(‘Hacked!’);*. У таблиці 2.1 приведені типові приклади санітізації вводу [23].

Валідація вводу – це перевірка, чи введені дані відповідають певним правилам чи критеріям. Це може бути перевірка на правильність формату (наприклад, електронної пошти або номера телефону) або перевірка на допустимі значення (наприклад, перевірка, чи вік користувача більше 18 років).

При реєстрації користувача валідація перевіряє, чи введений електронний адреса має правильний формат: user@example.com. Якщо адреса не має правильного формату електронної пошти, то виводиться помилка та користувач повинен виправити введені дані.

Таблиця 2.1 − Приклади санітації вводу

Небезпечний символ	**Санітація**
<script>	Заміна на *<script>*
</script>	Заміна на *</script>*
'; DELETE FROM users; --	Заміна на *'; DELETE FROM users; --*
<img src= ''x onerror= 'alert('Hacked! ') ''>	Заміна на *<img src= 'x ' onerror= 'alert('Hacked! ') '>*
Click me	Заміна на *Cli ck me*
<iframe src= 'https://malicious-site.com '></iframe>	*Заміна на <iframe src= 'https://malicious-site.com '></iframe>*
''><imgsrc=x onerror=alert('XSS')>	Заміна на *"><img src=x onerror=alert('XSS ')>*
*SELECT * FROM products WHERE name= ' '; DROP TABLE users; --*	Заміна на *SELECT * FROM products WHERE name= ' '; DROP TABLE users; --*
Clickme	Заміна на *Click me*

Санітізація та валідація вводу є важливими практиками для запобігання вразливостям вебсайтів. Вони допомагають забезпечити безпеку користувачів і захистити систему від атак. Для їх виконання можуть використовуватися спеціалізовані функції та бібліотеки, які дозволяють валідувати та санітувати дані перед їх обробкою.

2.2.2 Шифрування даних

Шифрування даних є одним із ключових методів захисту інформації на вебсайтах. Воно полягає в перетворенні звичайного тексту (під час передачі або зберігання) у формат, який стає незрозумілим для сторонніх осіб. Це робить дані безпечними, навіть якщо зловмисники отримають доступ до них.

Шифрування даних базується на математичних алгоритмах, які перетворюють звичайний текст у криптований (шифрований) вигляд. Щоб розкодувати дані, необхідно мати ключ розшифрування. Основні принципи включають:

– ключ шифрування: використовується ключ для шифрування та розшифрування даних. Це конфіденційна інформація, яку повинні знати лише вповноважені користувачі;

– симетричне та асиметричне шифрування: в симетричному шифруванні використовується один і той же ключ для шифрування та розшифрування [24]. У випадку асиметричного шифрування використовується пара ключів: публічний та приватний.

Існує кілька типів шифрування, включаючи:

– шифрування на рівні транспортування TLS/SSL: використовується для захисту передачі даних між клієнтом і сервером. Діє на рівні протоколу;

– шифрування на рівні зберігання: використовується для захисту даних під час їх зберігання на сервері або в базі даних;

– шифрування повного диска: захищає дані на рівні операційної системи, шифруючи весь диск або певні розділи;

– шифрування файлів та папок: забезпечує захист конкретних файлів або папок, незалежно від рівня операційної системи.

Переваги шифрування включають збереження конфіденційності даних, оскільки воно робить інформацію незрозумілою для незаконних користувачів. Крім того, воно забезпечує інтегрітет даних, допомагаючи виявляти будь-які незаконні зміни у інформації. Деякі методи шифрування також можуть включати процес аутентифікації користувачів, що дозволяє перевіряти їх легітимність та авторизувати доступ.

Однак шифрування має свої недоліки. Воно може бути витратним з точки зору обчислення, оскільки процес шифрування т розшифрування вимагає додаткових обчислень. Втрата ключів шифрування може призвести до повної втрати доступу до даних. Крім того, управління ключами шифрування може бути складним завданням, яке вимагає особливої уваги. Деякі методи шифрування також можуть мати обмеження на типи даних або їх обсяг.

2.2.3 Вебфайрволи (WAF)

Вебфайрволи є важливим елементом захисту вебсайтів та інформаційних ресурсів. Вони допомагають виявляти та блокувати небажані мережеві запити, що можуть бути атаками або спробами незаконного доступу.

Однією з переваг вебфайрволів є їх здатність аналізувати мережевий трафік і фільтрувати його в режимі реального часу. Вони можуть виявляти аномальну активність, таку як надмірні запити чи спроби використовувати вразливості вебдодатків. Це допомагає запобігати атакам, таким як DDoS або SQL injection.

Окрім цього, вебфайрволи можуть мати набори правил і фільтрів, які дозволяють налаштовувати доступ до ресурсів вебсайту (рис. 2.1). Це дозволяє обмежувати доступ до конфіденційної інформації лише авторизованим користувачам.

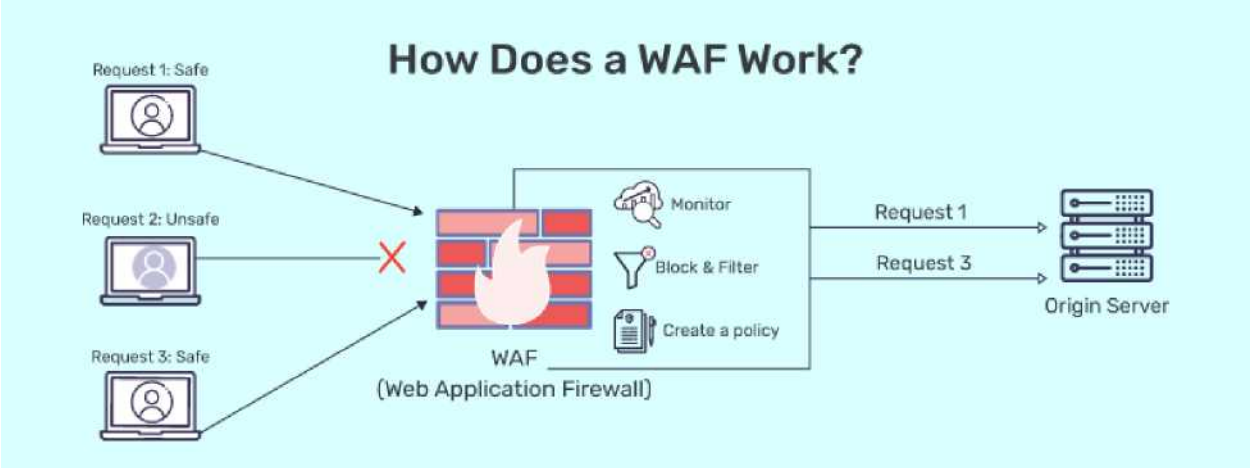

Рисунок 2.1 – Структура WAF

Проте вебфайрволи також мають свої недоліки. Вони можуть стати точкою витоку для атак, якщо не належним чином налаштовані. Крім того, вони можуть спричиняти затримки у роботі вебсайту через аналіз кожного запиту. Тому важливо налаштовувати вебфайрволи з урахуванням конкретних потреб і характеристик вебсайту.

2.2.4 HTTPS

HTTPS (Hypertext Transfer Protocol Secure) є фундаментальним елементом безпеки в інтернеті, який забезпечує захищене з'єднання між вебсервером та браузером. Це особливо важливо при передачі конфіденційних даних, таких як логіни, паролі та фінансова інформація.

HTTPS використовує протоколи SSL (Secure Sockets Layer) або TLS (Transport Layer Security) для шифрування даних, переданих між користувачем і вебсайтом. Це шифрування забезпечує конфіденційність і цілісність даних, захищаючи їх від перехоплення або змін зловмисниками.

Перехід на HTTPS підвищує довіру користувачів, оскільки вони можуть бачити, що їхні дані захищені. Крім того, це покращує позиціонування сайту в пошукових системах, оскільки багато пошукових систем, включаючи Google, вважають HTTPS одним із факторів ранжування.

Важливо регулярно оновлювати та управляти SSL/TLS сертифікатами. Прострочені сертифікати або сертифікати, видані ненадійними організаціями, можуть призвести до попереджень у браузерах користувачів і зменшити довіру до сайту.

HTTP Strict Transport Security (HSTS) – це політика безпеки, яка змушує браузери використовувати HTTPS для з'єднань з вебсайтом. Це допомагає запобігти атакам, які спробують переключити користувача на незашифроване з'єднання.

Старіші або застарілі алгоритми, такі як SSLv3 або ранні версії TLS, можуть бути вразливими до атак, тому вебсайти мають переконатися, що вони використовують останні та найбезпечніші версії цих протоколів.

2.2.5 Налаштування прав доступу до файлів та директорій

Правильне налаштування прав доступу до файлів та директорій є критично важливим для забезпечення безпеки вебсайту. Цей процес включає визначення та надання мінімально необхідних прав для різних користувачів і процесів, що взаємодіють з вебсервером [25]. Це допомагає запобігти несанкціонованому доступу або змінам у важливих файлах та директоріях.

Найважливіший принцип – це принцип найменших привілеїв, який полягає у наданні користувачам та процесам лише тих прав, які є абсолютно необхідними для їхньої роботи [26]. Наприклад, вебсерверу може знадобитися доступ на читання до більшості файлів сайту, але доступ на запис має бути обмежений лише для певних директорій

В Unix-подібних системах це часто реалізується через систему прав доступу, яка включає власника (owner), групу (group) та інших (others). Наприклад, права доступу 755 для директорії означають, що власник має право читання, запису та виконання, члени групи та інші користувачі – лише право читання та виконання.

Неправильно налаштовані права доступу можуть призвести до різних проблем безпеки, таких як несанкціоноване редагування файлів, виконання шкідливих скриптів або витоки даних. Наприклад, якщо для виконуваних файлів встановлено права доступу 777, будь-який користувач системи може змінити ці файли, що може призвести до компрометації сайту [27].

Рекомендується регулярно перевіряти права доступу до файлів та директорій, особливо після встановлення нових додатків або оновлень. Використання інструментів автоматизації та скриптів для моніторингу та управління правами доступу може значно підвищити рівень безпеки сайту.

2.2.5 Моніторинг активності та журналювання подій

Моніторинг активності та журналювання подій на вебсайті є ключовими компонентами в ідентифікації та реагуванні на потенційні безпекові загрози. Цей процес включає в себе збір, аналіз та зберігання інформації про всі дії, що відбуваються на сервері та в додатках. Це дозволяє адміністраторам вчасно виявляти несанкціоновані або підозрілі дії, такі як спроби неавторизованого доступу, зміни у файлах або аномалії в трафіку.

Інструменти моніторингу активності повинні бути налаштовані так, щоб забезпечувати комплексний огляд всієї системи. Це означає відстеження не тільки вебсерверів, але й баз даних, мережевого трафіку та інших критично важливих компонентів [28]. Наприклад, системи виявлення вторгнень (IDS) та системи управління подіями безпеки (SIEM) можуть бути використані для збору, аналізу та кореляції даних з різних джерел для виявлення потенційних загроз.

Журналювання подій відіграє важливу роль в процесі виявлення та розслідування інцидентів безпеки. Збереження детальних журналів дозволяє адміністраторам зрозуміти контекст та обставини безпекових інцидентів. Важливо, щоб журнали містили достатньо інформації для виявлення хто, коли та що зробив, а також які були наслідки цих дій [29].

Однак, ефективне журналювання вимагає не тільки збору даних, але й їх належного аналізу. Ручний перегляд журналів може бути трудомістким, тому важливо використовувати автоматизовані інструменти для моніторингу та аналізу. Ці інструменти можуть виявляти аномалії та автоматично сповіщати про підозрілі події.

Для забезпечення конфіденційності та цілісності журналів, необхідно також забезпечити їх належне зберігання. Журнали повинні зберігатися в безпечному місці з обмеженим доступом, щоб запобігти їх зміні або видаленню зловмисниками. Крім того, слід регулярно робити резервні копії журналів.

2.2.5 Використання сильних паролів та двофакторної аутентифікації

Використання сильних паролів та двофакторної аутентифікації є ключовими елементами у забезпеченні безпеки вебсайтів. Сильні паролі складаються з великої кількості символів, уключаючи цифри, великі та малі літери, а також спеціальні символи [30]. Це значно ускладнює завдання зловмисникам, які намагаються зламати пароль за допомогою методів грубої сили або вгадування (рис. 2.2).

Рисунок 2.2 – Складність Brute Force

Двофакторна аутентифікація (2FA) додає додатковий рівень захисту, вимагаючи від користувачів ввести не тільки свій пароль, але й другий фактор, яким може бути код, отриманий через SMS або електронну пошту, або ж генерований спеціальним додатком. Це значно знижує ризик несанкціонованого доступу, навіть якщо пароль був скомпрометований. Для ілюстрації ефективності 2FA можна використати діаграму, яка демонструє

зниження випадків несанкціонованого доступу після впровадження 2FA [31].

Важливо також розробити та дотримуватися політики сильних паролів. Це включає в себе вимоги до мінімальної довжини пароля, його складності та регулярну зміну паролів. Така політика повинна бути чітко сформульована і доступна всім користувачам системи. Інформаційний бюлетень або інфографіка з кращими практиками створення сильних паролів може бути корисною у вихованні користувачів.

Окрім цього, слід розглянути можливість використання менеджерів паролів. Менеджери паролів дозволяють користувачам зберігати та керувати своїми паролями в безпечному середовищі. Вони також можуть генерувати сильні, випадкові паролі, що забезпечують додатковий рівень безпеки.

2.2.6 Регулярне оновлення програмного забезпечення та вразливих компонентів

Регулярне оновлення програмного забезпечення та вразливих компонентів є критично важливим для забезпечення безпеки вебсайтів. Це включає в себе оновлення операційної системи, вебсерверного програмного забезпечення, баз даних, CMS (систем управління контентом), плагінів та будь-якого іншого стороннього ПЗ, яке використовується на сайті. Оновлення часто містять виправлення вразливостей, які могли бути виявлені від моменту останнього релізу, тим самим знижуючи ризик кібератак.

Регулярне оновлення програмного забезпечення та вразливих компонентів є критично важливим для забезпечення безпеки вебсайтів. Це включає в себе оновлення операційної системи, вебсерверного програмного забезпечення, баз даних, CMS (систем управління контентом), плагінів та будь-якого іншого стороннього ПЗ, яке використовується на сайті. Оновлення часто містять виправлення вразливостей, які могли бути виявлені від моменту останнього релізу, тим самим знижуючи ризик кібератак.

Неоновлене ПЗ може бути серйозним ризиком безпеки. Зловмисники часто використовують відомі вразливості в застарілому програмному

забезпеченні для проведення атак, таких як SQL ін’єкції, кросс-сайтовий скриптинг (XSS) та інші. Оновлення ПЗ також може включати в себе вдосконалення функціональності та продуктивності, що також важливо для підтримки ефективної роботи вебсайту [32]. Також важливо проводити тестування після впровадження оновлень, щоб переконатися, що нові версії не спричиняють проблем зі стабільністю або сумісністю на сайті. Чек-лист для тестування оновлень може включати перевірку ключових функцій сайту, сумісність з іншими компонентами та відсутність нових помилок.

2.2.7 Регулярне оновлення програмного забезпечення та вразливих компонентів

Аудит безпеки та пенетраційне тестування є важливими елементами у підтримці безпеки вебсайтів. Аудит безпеки включає в себе всебічний аналіз поточного стану безпеки сайту, виявляючи потенційні слабкості та вразливості. Цей процес може включати перевірку конфігурацій серверів, політик безпеки, прав доступу, а також використання захисних технологій.

Пенетраційне тестування (часто згадуване як «pen testing») є процесом активного тестування системи на предмет вразливостей, які можуть бути використані зловмисниками. Це включає в себе симуляцію кібератак на систему з метою виявлення та усунення безпекових прорахунків. Пенетраційне тестування повинно проводитися кваліфікованими фахівцями, які використовують різноманітні методи та інструменти для імітації реальних атак.

Після проведення пенетраційного тестування, важливо ретельно проаналізувати отримані результати та розробити план дій для усунення виявлених вразливостей [33]. Це може включати в себе зміни в конфігурації системи, впровадження додаткових захисних механізмів або навчання персоналу.

2.2.8 Техніки захисту від CSRF

Захист від міжсайтової фальсифікації запитів (CSRF) охоплює кілька технічних реалізацій. Одним із найефективніших методів є використання унікальних токенів CSRF, які генеруються сервером для кожної сесії користувача. Ці токени вбудовуються в HTML форми та перевіряються сервером при кожному запиті, що забезпечує, що запит дійсно надійшов від легітимного користувача.

Ще одним важливим елементом захисту є використання атрибуту SameSite для кукі. Цей атрибут контролює, чи будуть куки надсилатися при міжсайтових запитах. Наприклад, встановлення SameSite=Lax або SameSite=Strict дозволяє запобігти використанню куків у міжсайтових запитах, що складніше для потенційних атак CSRF.Перевірка походження запиту через HTTP-заголовки, такі як Referer та Origin, також є важливою технікою. Сервери можуть використовувати ці заголовки для підтвердження, що запит прийшов з довіреного джерела, і таким чином блокувати запити, що не відповідають очікуваному джерелу.

Додатково, застосування HTTP-заголовків для безпеки, як-от X-Frame-Options, може допомогти у запобіганні атак «clickjacking», які є однією з форм CSRF. Цей заголовок блокує вбудовування сторінок сайту в рамки (iframes) на інших сайтах, ускладнюючи виконання атак.

Нарешті, обмеження часу дії сесійних токенів та куків є ще одним методом зменшення ризиків пов’язаних з CSRF. Це забезпечує, що навіть якщо токен або куки будуть перехоплені, їхній час дії буде обмежений, що мінімізує можливість їх зловмисного використання.

2.2.9 Параметризовані запити

Параметризовані запити є однією з основних технік захисту від SQL ін’єкцій (SQLi). Вони полягають у розділенні SQL команд від вхідних даних користувача, що запобігає інтерпретації вхідних даних як частини SQL коду. Цей підхід використовується для створення запитів до бази даних, де користувацький ввід обробляється окремо і не може змінити логіку самого запиту.

Зазвичай, у параметризованих запитах використовуються спеціальні маркери або плейсхолдери для вхідних даних, наприклад, ? або @name у SQL командах. Коли запит виконується, ці плейсхолдери замінюються реальними даними, які були правильно оброблені і екрановані, щоб запобігти інтерпретації їх як SQL коду.

Прикладом може бути запит на вибірку з бази даних, де замість безпосереднього вставляння користувацького вводу у рядок запиту, ввод вставляється в запит через параметри. Наприклад, замість SQL команди *SELECT * FROM users WHERE username = ‘ “ + username + ’’ ’,* використовується безпечний варіант: *SELECT * FROM users WHERE username = ?,* де *username* передається як параметр.

2.2.10 Content Security Policy (CSP)

Використання політик Content Security Policy (CSP) є ефективним способом захисту від кросс-сайтового скриптингу (XSS). CSP дозволяє вебсайтам визначати, з яких джерел може бути завантажений контент, тим самим обмежуючи можливості для виконання шкідливих скриптів.

CSP реалізується через встановлення відповідних HTTP-заголовків, які вказують браузерам, як обробляти контент на сторінці. Наприклад, політика може заборонити завантаження скриптів, стилів або зображень з будь-яких джерел, крім тих, що виразно дозволені.

Основні аспекти використання CSP включають:

– обмеження зовнішніх ресурсів: CSP може вказати, що скрипти можуть завантажуватися лише з певних довірених джерел, наприклад, з того ж домену, що й основна сторінка або з конкретних доменів, яким ви довіряєте;

– заборона інлайн-скриптів: політика може заборонити використання інлайн-скриптів (скриптів, що розміщені прямо в HTML-документі), що є поширеним вектором для XSS-атак;

– запобігання використанню eval() та інших небезпечних функцій: CSP може заборонити використання JavaScript-функцій, які можуть виконувати код, переданий як рядок, наприклад, eval();

– використання Nonce або Hashes для скриптів: CSP дозволяє використовувати nonce (унікальний токен для кожного запиту) або хеші для ідентифікації легітимних скриптів, тим самим запобігаючи виконанню недозволених скриптів [34].

2.2.11 Використання безпечних шаблонних движків для уникнення SSTI

Використання безпечних шаблонних движків є ключовим елементом у запобіганні шаблонним ін’єкціям сервера (Server-Side Template Injection, SSTI). Ця техніка полягає в обмеженні можливості вставки або виконання шкідливого коду через шаблони, які використовуються для генерації динамічного контенту на вебсайтах.

Безпечні шаблонні движки автоматично санітізують ввод, що дозволяє запобігти інтерпретації користувацького вводу як частини коду шаблону. Наприклад, движки, такі як Jinja2 у Python або Liquid у Ruby, мають вбудовані механізми, які виявляють і блокують спроби вставки шкідливих скриптів через шаблони.

Один із прикладів – якщо в шаблон передається змінна, що містить користувацький ввод, безпечний шаблонний движок автоматично екранує спеціальні символи, які можуть бути інтерпретовані як частина команди або виразу в шаблоні. Таким чином, навіть якщо ввод містить потенційно шкідливі конструкції, вони будуть оброблені як звичайний текст, а не як код. Використання безпечних шаблонних движків є ключовим елементом у

запобіганні шаблонним ін'єкціям сервера (Server-Side Template Injection, SSTI). Ця техніка полягає в обмеженні можливості вставки або виконання шкідливого коду через шаблони, які використовуються для генерації динамічного контенту на вебсайтах (рис.2.3).

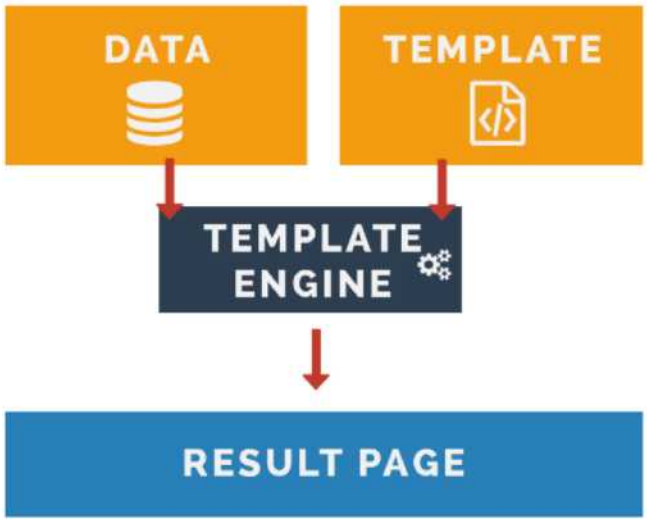

Рисунок 2.3 – Концепція Template Engine

Безпечні шаблонні движки автоматично санітізують ввод, що дозволяє запобігти інтерпретації користувацького вводу як частини коду шаблону. Наприклад, движки, такі як Jinja2 у Python або Liquid у Ruby, мають вбудовані механізми, які виявляють і блокують спроби вставки шкідливих скриптів через шаблони.

Один із прикладів – якщо в шаблон передається змінна, що містить користувацький ввод, безпечний шаблонний движок автоматично екранує спеціальні символи, які можуть бути інтерпретовані як частина команди або виразу в шаблоні. Таким чином, навіть якщо ввод містить потенційно шкідливі конструкції, вони будуть оброблені як звичайний текст, а не як код.

Крім того, деякі шаблонні движки дозволяють використовувати більш строгі політики безпеки, які можуть повністю відключити можливість виконання певних типів виразів або команд в шаблонах. Це може бути корисним у середовищах, де високі вимоги до безпеки.

Важливо відзначити, що вибір безпечного шаблонного движка та його правильна конфігурація є критичними для запобігання SSTI [35]. Розробники повинні бути обізнані з можливостями і налаштуваннями обраного движка,

щоб максимально ефективно використовувати його безпекові можливості.

2.3 Типові атаки та приклади впровадження захисту

У сфері електронної комерції часто стикаються з SQL ін’єкціями, що ставить під загрозу безпеку баз даних. Один з популярних інтернет-магазинів вирішив цю проблему, впровадивши параметризовані запити. Це дозволило ізолювати вхідні дані користувачів від SQL команд, значно підвищивши безпеку даних. Додатково була встановлена система моніторингу для виявлення та реагування на спроби несанкціонованого доступу.

В банківській сфері, де високий ризик фішингових атак, один з провідних банків запровадив обов’язкову двофакторну аутентифікацію для всіх онлайн-транзакцій. Це включало надсилання одноразового коду на мобільний телефон користувача, що значно знизило ризик несанкціонованого доступу до банківських рахунків.

Соціальні мережі, стикаючись з викликами XSS атак, активно використовують Content Security Policy (CSP). Один із великих соціальних мереж ввів строгі політики CSP, які обмежували завантаження скриптів лише з довірених джерел. Такий підхід допоміг запобігти виконанню потенційно шкідливих скриптів, що могли бути вставлені через користувацькі коментарі.

В освітніх установах часто виникають випадки втручання у конфіденційність даних. Один з університетів посилив захист своєї інформаційної системи, впровадивши шифрування даних та використовуючи більш сучасні брандмауери та антивірусні рішення. Це допомогло захистити персональні дані студентів та персоналу.

Урядові та державні сайти, які часто стають мішенями цілеспрямованих кібератак, посилили свої захисні механізми, включаючи розширений моніторинг мережевого трафіку та впровадження складних протоколів аутентифікації. Такі кроки допомогли підвищити безпеку критично важливих державних ресурсів та захистити від потенційних кіберзагроз.

У сфері здоров’я, де безпека пацієнтських даних є високим пріоритетом,

одна з лікарень впровадила розширені механізми шифрування даних та обмеження доступу до пацієнтських записів. Це допомогло запобігти несанкціонованому доступу до чутливої медичної інформації, а також забезпечило відповідність вимогам щодо конфіденційності пацієнтів.

У галузі технологій, де ризик DDoS-атак є високим, одна технологічна компанія впровадила розширені системи захисту від DDoS. Це включало використання масштабованих мережевих фільтрів та розподіленої системи виявлення та нейтралізації атак, які допомогли забезпечити стабільність та доступність їх онлайн-сервісів [36].

Для туристичних агентств та готельного бізнесу, де часто виникають випадки шахрайства з кредитними картками, було важливо впровадити системи безпеки, що забезпечують безпечну обробку платіжних даних. Використання стандарту PCI DSS допомогло гарантувати, що всі транзакції обробляються, зберігаються та передаються у безпечному середовищі, мінімізуючи ризики витоку фінансової інформації.

У сфері онлайн-освіти, де важливим є захист інтелектуальної власності та персональних даних, одна освітня платформа впровадила розширені системи контролю доступу та аутентифікації. Це забезпечило, що доступ до освітніх матеріалів та студентських даних мають тільки уповноважені особи, що значно знизило ризик неавторизованого розповсюдження контенту.

Для медіа-порталів та новинних сайтів, які часто стають ціллю для атак на свободу слова, важливим є захист від цензури та блокування контенту. Один із відомих новинних порталів впровадив рішення для анонімізації трафіку та обходу блокувань, забезпечуючи доступність свого контенту для читачів у регіонах з обмеженим доступом до інформації.

2.4 Системи виявлення та попередження вторгнень (IDS/IPS)

2.4.1 IDS

Системи виявлення вторгнень (Intrusion Detection Systems, IDS) є критичним компонентом у сфері кібербезпеки, оскільки вони забезпечують моніторинг мережі та систем на предмет підозрілої або шкідливої активності. IDS можна класифікувати за різними критеріями, залежно від їх функціональності та способу впровадження.

Одним із ключових аспектів роботи IDS є здатність аналізувати мережевий трафік та системні журнали для виявлення ознак вторгнення. Це може включати в себе виявлення відомих підписів атак, таких як віруси, трояни, або аналіз поведінки для виявлення потенційно шкідливих або ненормальних дій. Системи, засновані на підписах, залежать від баз даних відомих шаблонів атак, тоді як системи, засновані на аналізі поведінки, намагаються ідентифікувати аномалії, які можуть вказувати на нові або невідомі атаки.

Щодо класифікації, IDS поділяються на мережеві (NIDS) та хост-базовані (HIDS) системи. Мережеві IDS встановлюються на стратегічних точках у мережі для моніторингу трафіку, що проходить через мережу. Наприклад, NIDS може бути розміщений на кордоні між корпоративною мережею та Інтернетом, щоб відстежувати вхідний і вихідний трафік. Натомість, хост-базовані IDS встановлюються на окремих серверах або робочих станціях і аналізують системні журнали та активність на цих хо стах, виявляючи зміни в системних файлах або ненормальну активність процесів.

Ще одним важливим аспектом IDS є спосіб реагування на виявлені загрози. Деякі системи просто реєструють або сповіщають про підозрілі події, тоді як інші можуть взаємодіяти з мережевим обладнанням, щоб блокувати шкідливий трафік або ізолювати заражені хости.

Системи виявлення вторгнень постійно розвиваються, адаптуючись до нових викликів у кібербезпеці. Наприклад, з розвитком хмарних технологій та розподілених систем, з’явилися хмарні IDS, які можуть виявляти атаки у хмарних середовищах. Ці системи забезпечують моніторинг хмарних ресурсів,

аналізуючи логи та трафік між хмарними сервісами.

Серед численних систем виявлення вторгнень (IDS) на ринку, деякі з них виділяються своєю ефективністю та популярністю. Розглянемо три такі системи і проаналізуємо їхню ефективність:

– Snort: є однією з найбільш відомих і широко використовуваних мережевих IDS. Ця система працює на основі підписів для виявлення відомих шаблонів атак. Snort може бути налаштований для виконання як в пасивному режимі, так і в активному, використовуючи вбудовані механізми для блокування шкідливого трафіку. Ефективність Snort полягає в його гнучкості та великій базі даних підписів, яка регулярно оновлюється. Однак, він може бути менш ефективним проти невідомих або складно маскуваних атак, що не мають визначених підписів;

– Suricata: є ще однією мережевою IDS, яка забезпечує високу продуктивність і підтримку сучасних мережевих протоколів. Вона може використовувати як аналіз підписів, так і виявлення аномалій. Suricata також забезпечує додаткові функції, такі як підтримка многопотоковості та автоматичне виявлення застосунків. Ця IDS особливо ефективна в багатопотокових та високонавантажених мережевих середовищах завдяки своїм оптимізованим алгоритмам. Однак, вона може вимагати більш складної налаштування та ресурсів для ефективної роботи;

– OSSEC: є прикладом хост-базованої IDS, яка зосереджена на моніторингу окремих серверів або пристроїв. Вона аналізує логи системи, виявляє зміни у файловій системі та перевіряє цілісність важливих файлів. Ефективність OSSEC виражається у її здатності виявляти внутрішні спроби вторгнення та невідповідності в системі. Ця система є корисною для глибокого моніторингу конкретних хостів, але може вимагати значних зусиль для налаштування та управління.

2.4.2 IPS

Системи запобігання вторгненням (Intrusion Prevention Systems, IPS) являють собою ключовий елемент інфраструктури кібербезпеки, надаючи активний захист проти різноманітних кібератак. IPS, на відміну від систем виявлення вторгнень (IDS), не лише виявляють потенційні загрози, але й активно втручаються, щоб запобігти або зменшити шкоду від атак.

Принципи роботи IPS базуються на глибокому аналізі мережевого трафіку та активності системи для ідентифікації ознак вторгнення або шкідливих дій. Це може включати в себе виявлення відомих підписів атак, аналіз поведінкових патернів та використання різноманітних алгоритмів машинного навчання для ідентифікації аномальної поведінки.

У рамках класифікації, IPS можуть бути розділені на кілька типів:

– мережеві IPS (NIPS): встановлюються на стратегічних точках у мережі і моніторять та аналізують весь мережевий трафік. Вони здатні блокувати атаки, які виявлені у мережі, такі як DDoS атаки, сканування портів, атаки на застосунки тощо;

– хост-базові IPS (HIPS): хост-базовані системи встановлюються безпосередньо на серверах або робочих станціях і забезпечують захист від атак, спрямованих на конкретні хости. Вони можуть ідентифікувати та блокувати шкідливі процеси, зміни в системних файлах та інші ознаки компрометації;

– бездротові IPS (WIPS): спеціалізовані на захисті бездротових мереж, WIPS забезпечують виявлення та блокування атак, специфічних для Wi-Fi мереж, таких як несанкціонований доступ або «man-in-the-middle» атаки;хмарні IPS (CIPS): ці системи призначені для захисту хмарних середовищ та можуть бути інтегровані з хмарними службами для моніторингу та захисту хмарної інфраструктури та додатків.

Серед різноманітних систем запобігання вторгненням (IPS) на ринку, деякі з них виділяються своєю надійністю та ефективністю. Ось приклади чотирьох таких систем:

– Cisco Firepower;

- Palo Alto Networks IPS;
- Fortinet FortiGate;
- IBM Security Network IPS.

Cisco Firepower є однією з передових IPS, яка поєднує в собі функціональність мережевого і хост-базованого захисту. Вона використовує глибокий аналіз пакетів та розширену обробку даних для виявлення та блокування атак. Firepower також інтегрується з іншими продуктами Cisco для забезпечення єдиного підходу до безпеки. Її ефективність полягає у високій точності виявлення загроз та мінімальній кількості помилкових спрацьовувань, але вона може вимагати складної налаштування та управління [37].

Palo Alto Networks пропонує IPS з сильними функціями аналізу та адаптивної безпеки. Ця система може аналізувати широкий спектр трафіку, включаючи зашифровані з'єднання, і забезпечує детальний аналіз для виявлення складних атак. Ефективність системи Palo Alto полягає у її здатності адаптуватися до нових загроз та швидко реагувати на них. Однак, висока вартість та складність можуть бути бар'єром для деяких організацій [38].

FortiGate від Fortinet є іншим популярним рішенням IPS, яке інтегрується з широким портфоліо безпеки Fortinet. Ця система забезпечує ефективне виявлення та блокування атак, а також можливість налаштування правил безпеки. FortiGate відомий своєю високою продуктивністю та ефективністю у виявленні та блокуванні атак. Але, як і більшість комплексних систем, вимагає відповідної експертизи для оптимального конфігурування.

IBM пропонує IPS, який використовує передові технології, такі як штучний інтелект і машинне навчання, для виявлення та реагування на атаки. Система забезпечує розширений аналіз трафіку та ефективно ідентифікує складні загрози. IBM Security Network IPS є ефективним у виявленні різноманітних загроз, включаючи нові та адаптивні атаки. Однак, високий рівень складності та вартість можуть бути перепоною для малих та середніх підприємств.

У цілому, ефективність IPS залежить від правильного вибору системи, яка відповідає специфіці мережевого середовища та потребам організації.

Регулярне оновлення, налаштування та адаптація до змінюваного кіберландшафту є ключовими для підтримки ефективності IPS.

2.5 Оцінка ризиків управління безпекою вебсайтів

Оцінка ризиків управління безпекою вебсайтів є фундаментальним процесом, який дозволяє організаціям ідентифікувати, аналізувати та керувати потенційними загрозами для їх онлайн-ресурсів. Цей процес включає в себе кілька ключових кроків, які допомагають визначити вразливі місця та розробити стратегії для мінімізації ризиків.

Першим кроком є ідентифікація активів. Це означає визначення всіх компонентів вебсайту, які потребують захисту – від фізичних серверів та мережевого обладнання до програмного забезпечення та даних користувачів. Важливо також враховувати різні типи даних, які обробляються та зберігаються на вебсайті, оцінюючи їхню чутливість та значимість для бізнесу.

Наступний крок полягає у визначенні потенційних загроз. Це включає в себе аналіз можливих джерел атак, таких як хакери, шкідливе програмне забезпечення, фішинг, а також внутрішні загрози, як-от помилки персоналу або ненадійні співробітники. Важливо також враховувати зовнішні фактори, як-от технологічні зміни, законодавчі вимоги та еволюцію кіберзагроз.

Після ідентифікації активів та загроз проводиться аналіз ризиків. Це процес оцінки ймовірності та потенційних наслідків різних видів атак або інцидентів. Аналіз ризиків допомагає встановити пріоритетність заходів безпеки, заснованих на балансі між вартістю захисту та потенційними втратами від інцидентів [39].

Четвертим кроком є розробка стратегії управління ризиками. Це може включати в себе різні заходи, такі як посилення безпеки інфраструктури, впровадження політик безпеки, регулярне навчання персоналу та створення плану реагування на інциденти [40]. Важливо також регулярно оновлювати заходи безпеки, щоб вони відповідали змінюваному ландшафту загроз.

На останньому етапі проводиться моніторинг та перегляд системи

управління ризиками. Це включає в себе регулярну перевірку ефективності заходів безпеки, оцінку нових загроз та адаптацію стратегії управління ризиками відповідно до змін у бізнес-процесах та технологіях. Регулярний моніторинг забезпечує, що система безпеки залишається ефективною у протистоянні з ризиками, пов’язаними з вебсайтом.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ СКАНЕРА

3.1 Обґрунтування вибору середовища програмної реалізації

У процесі програмної реалізації сканера вразливостей вебсайту була обрана мова програмування Python. Python є однією з найпопулярніших мов програмування, відомою своєю читабельністю, гнучкістю та широким спектром застосування. Це високорівнева мова, що дозволяє розробникам ефективно писати надійний код, витрачаючи менше часу на технічні деталі нижчого рівня.

Використання специфічних бібліотек Python, таких як requests, logging, selenium, beautifulsoup4 та colorama, дозволяє ефективно вирішувати задачі, пов’язані з вебскануванням та аналізом даних:

– Requests є бібліотекою для здійснення HTTP-запитів, що є необхідним для інтеракції з вебсайтами та отримання відповідей від сервера;

– Logging забезпечує ведення журналів процесу сканування, що є важливим для відстеження ходу виконання програми та діагностики можливих проблем;

– Selenium використовується для автоматизації веббраузерів, що дозволяє сканеру взаємодіяти з вебсайтами, як це робить звичайний користувач. Це особливо важливо для тестування на DOM-based XSS, де необхідно виконувати скрипти в контексті вебсторінки;

– BeautifulSoup4 спрощує процес парсингу HTML та XML документів, що дозволяє ефективно аналізувати структуру вебсайтів та витягувати необхідну інформацію.

Colorama використовується для додавання кольору до тексту в консолі, що робить вивід програми більш читабельним та допомагає визначити важливі частини звіту сканування.

Python разом з цими бібліотеками створює ефективне середовище для розробки сканера вразливостей, здатного ефективно виконувати комплексні завдання аналізу безпеки вебсайтів.

3.2 Програмна реалізація

Програма реалізована в інтерфейсі Command Line Interface (CLI) у вигляді Python скрипту. Реалізовані методи сканування на SQL Injection (без порушення цілісності та прив'язки до назв), Reflected XSS, DOM Based XSS, Remote Code Execution, Server-Side Template Injection, Open Redirection, Clickjacking, тестування API. Також реалізована функція проходження повного переліку тесту за вказаною глибиною відносно URL.

Проєкт для тестування розташований на localhost. Він представляє з себе вебдодаток з навмисно проробленими вразливостями до SQLi та Reflected XSS для пентестингу з публічного репозиторію Github (dvwpa).

В першу чергу, було реалізована аутентифікація на вебсайті, а далі сканування виконується в сесії.

Були створені payloads для сканування на вразливості до SQLi, XSS, Open Redirect, SSTi, RCE.

Також реалізована багатопоточність для більшої швидкості відправки запитів, хоч Python не може працювати на декількох ядрах.

Знайдені вразливості логуються у файл.

3.3 Інструкція користувача

Програмний інтерфейс розроблено таким чином, щоб забезпечити простоту та інтуїтивність використання:

Крок 1. Перейти до каталогу vuln_scan, встановити бібліотеки.

Крок 2. Запустити main.py.

Крок 3. Ввести дані входу та посилання для авторизації.

Крок 4. Обрати один з запропонованих варіанті сканування.

Крок 5. Обрати метод для певних типів сканування.

Крок 6. Ввести посилання на сторінку для сканування.

Крок 7. Результат буде виведений в консоль та у файл.

3.4 Тестування програми

Було протестовано SQLi, XSS, RCE, SSTi, Open Redirection, Clickjacking.

В результаті було виявлено, що вебдодаток має вразливості до Reflected/Stored XSS, SQLi та не вразливий до RCE, SSTi, Clickjacking, Open Redirect.

На рисунках 3.1 та 3.2 показана аутентифікація та сканування сайту на вразливість до Stored/Reflected XSS.

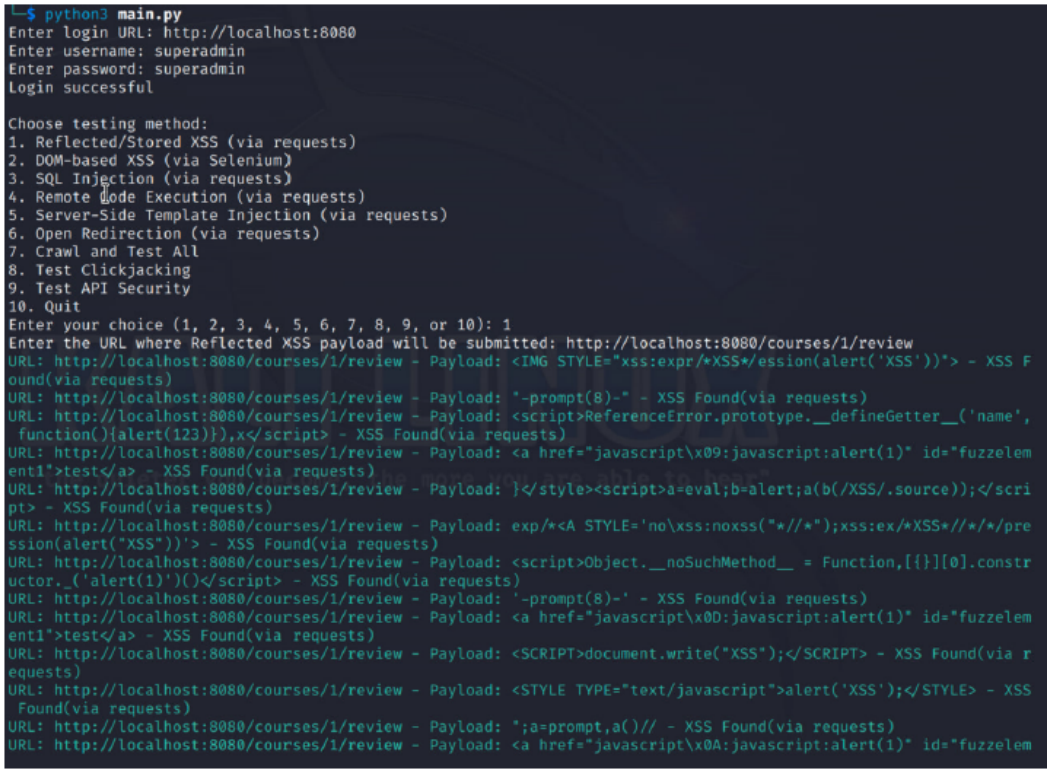

Рисунок 3.1 – Аутентифікація та сканування сайту на вразливість до Stored/Reflected XSS.

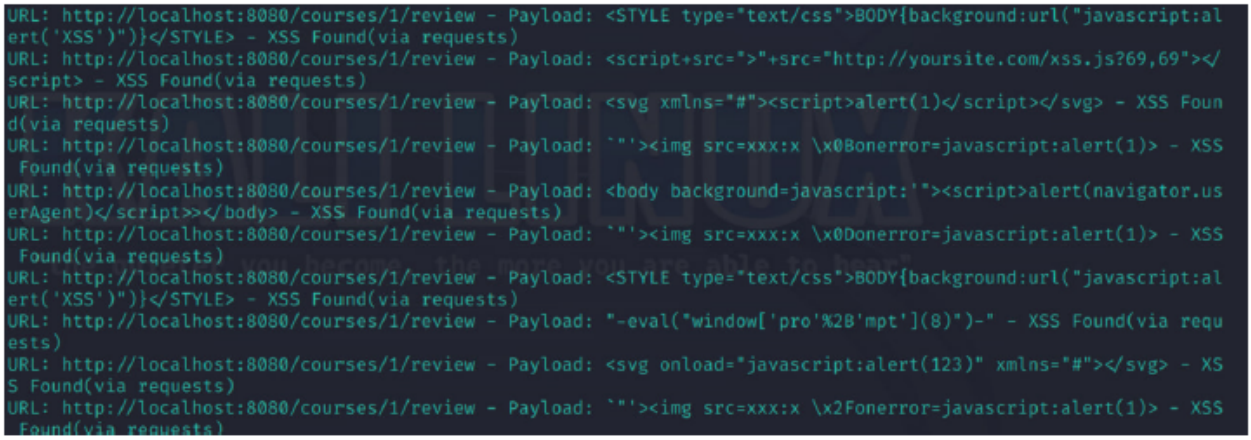

Рисунок 3.2 – Результат сканування сайту на вразливість до Stored/Reflected XSS

На рисунках 3.3 та 3.4 продемонстровано сканування сайту на вразливість до SQL Injection через форму.

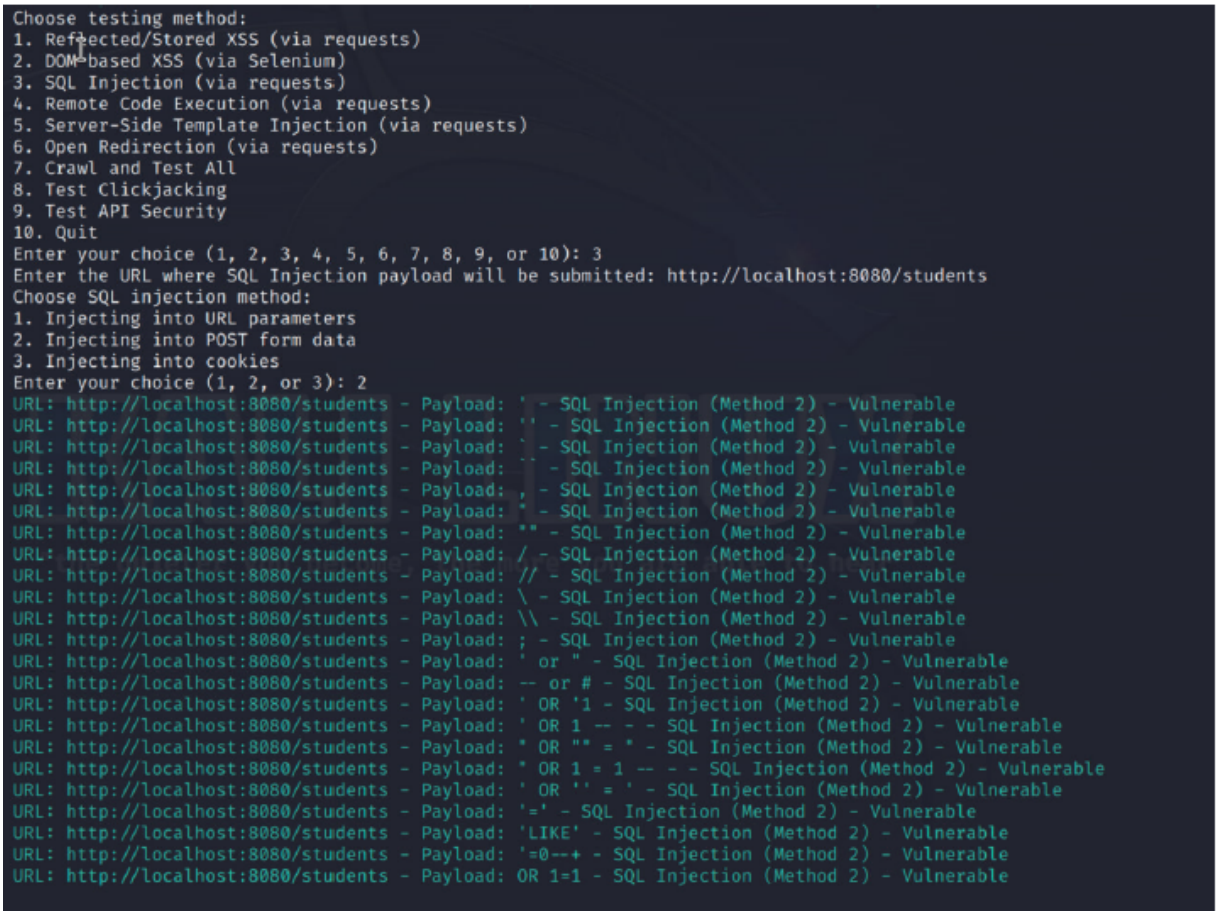

Рисунок 3.3 – Сканування сайту на SQLi через форму.

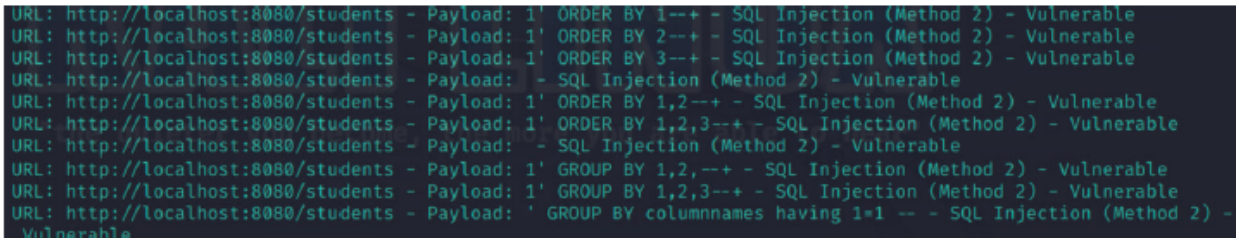

Рисунок 3.4 – Сканування сайту на SQLi через форму.

На рисунках 3.5, 3.6, 3.7 продемонстровано сканування сайту на вразливість до RCE через форму.

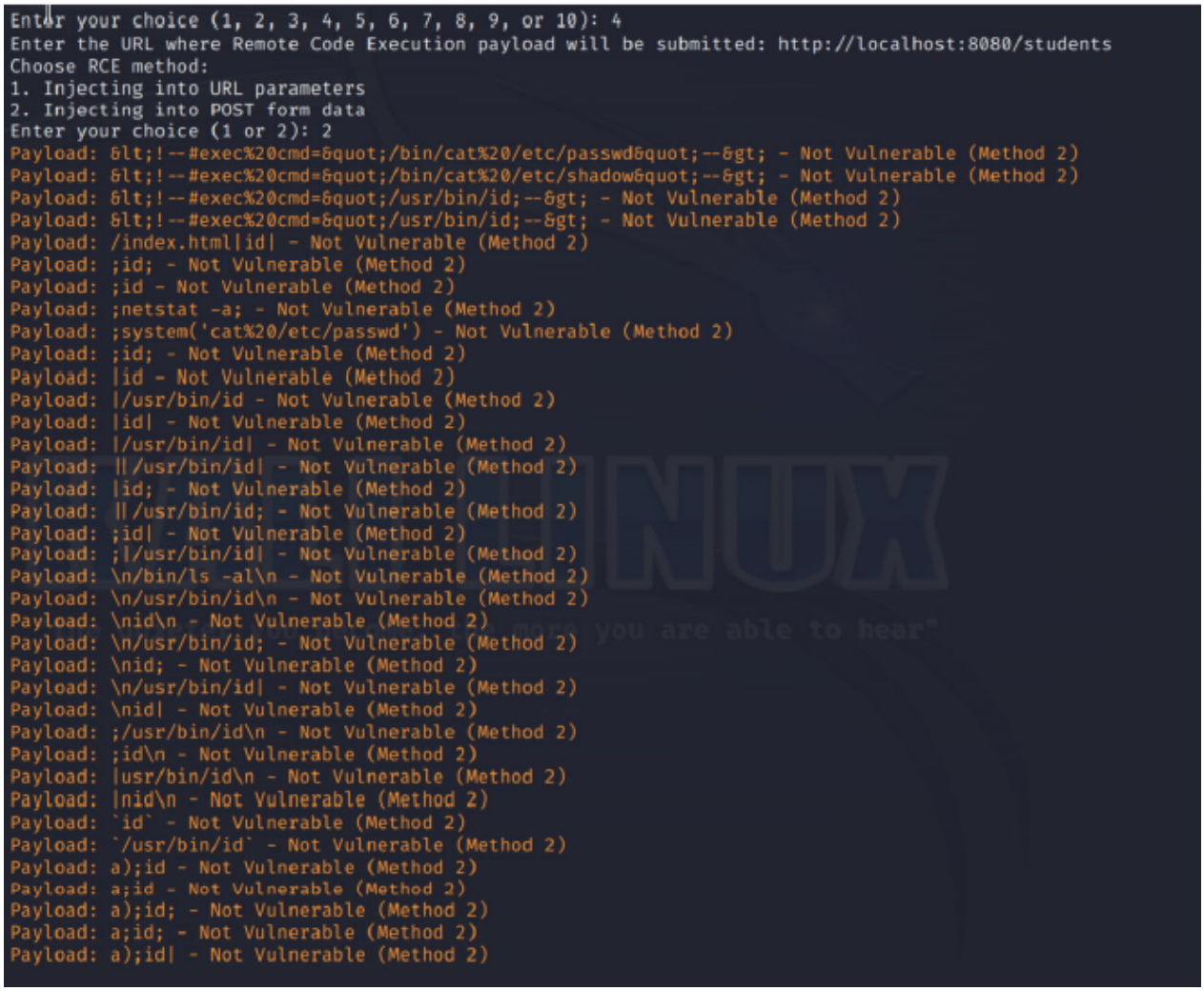

Рисунок 3.5 – Сканування сайту на вразливість до RCE через форму

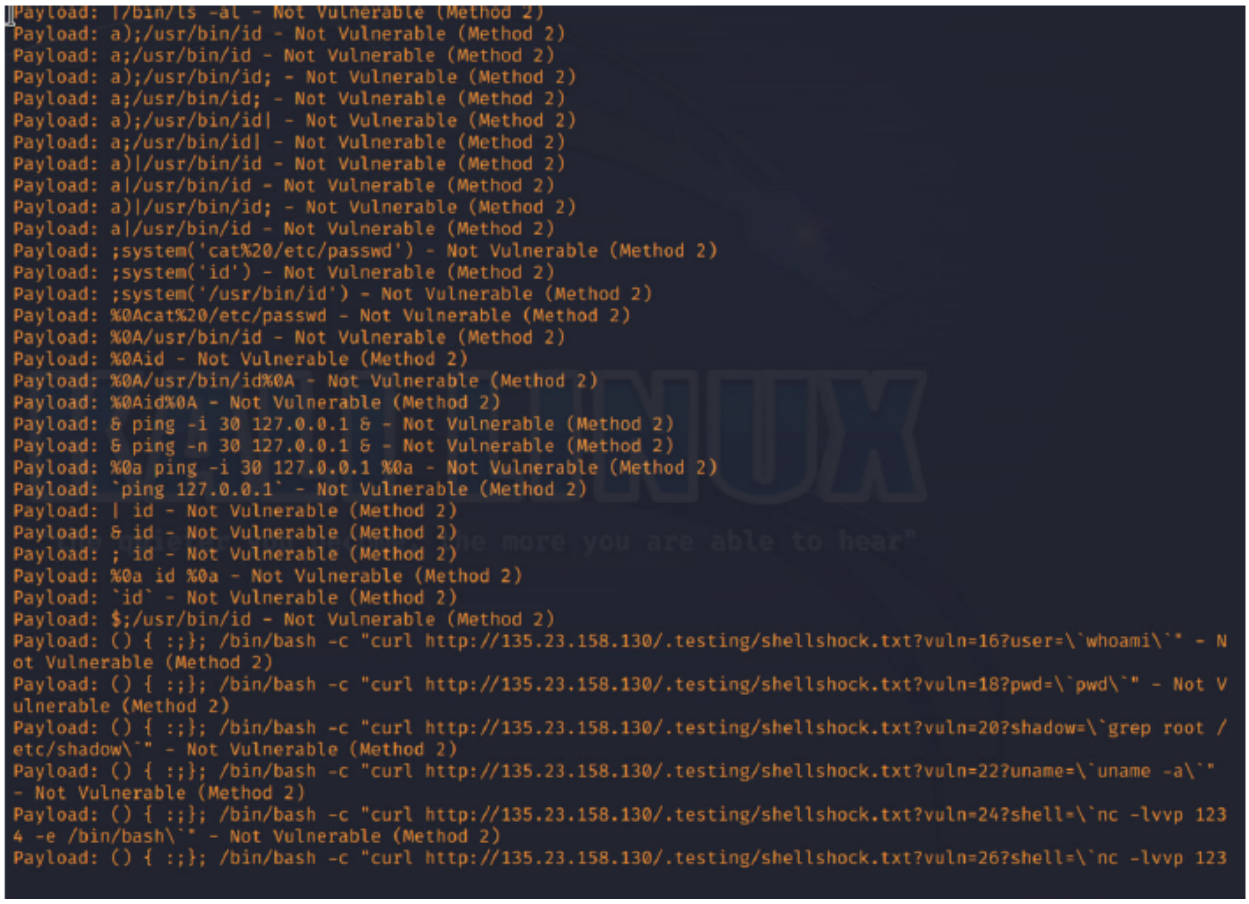

Рисунок 3.6 – Сканування сайту на вразливість до RCE через форму.

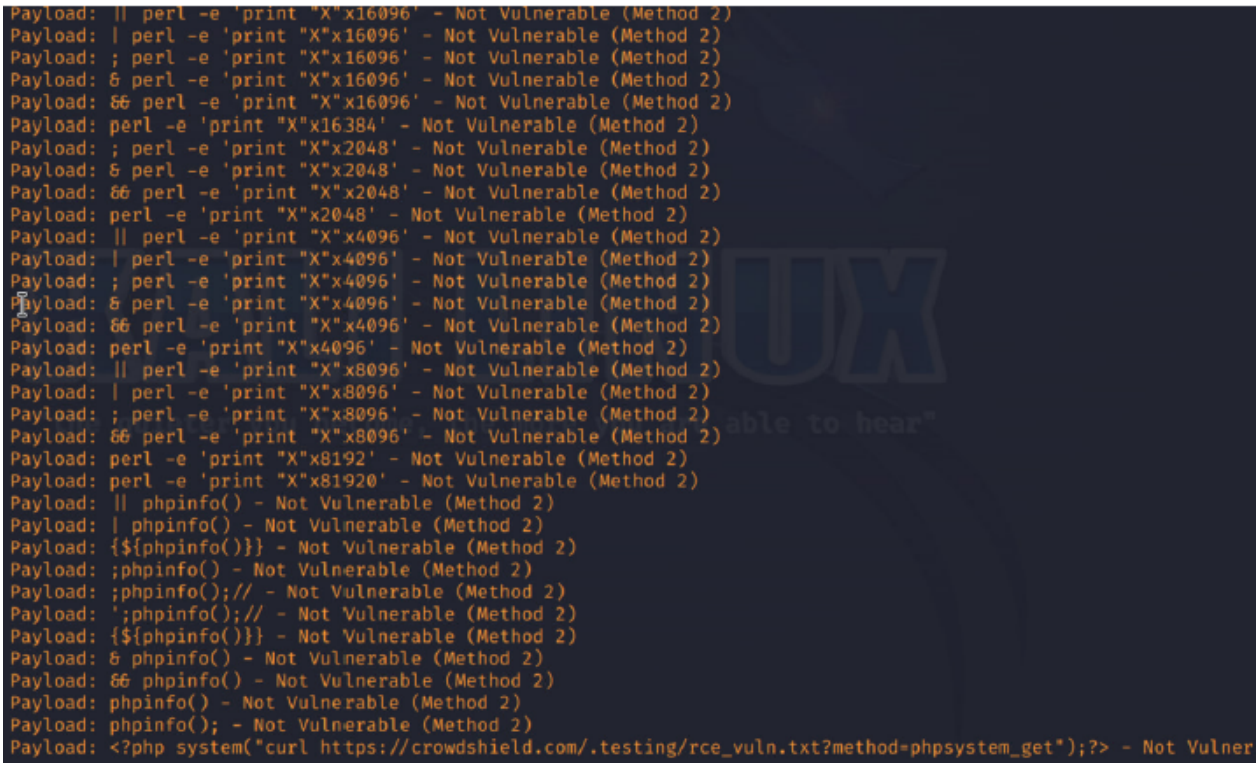

Рисунок 3.7 – Сканування сайту на вразливість до RCE через форму.

На рисунках 3.8 та 3.9 продемонстровано сканування сайту на

вразливість до SSTi.

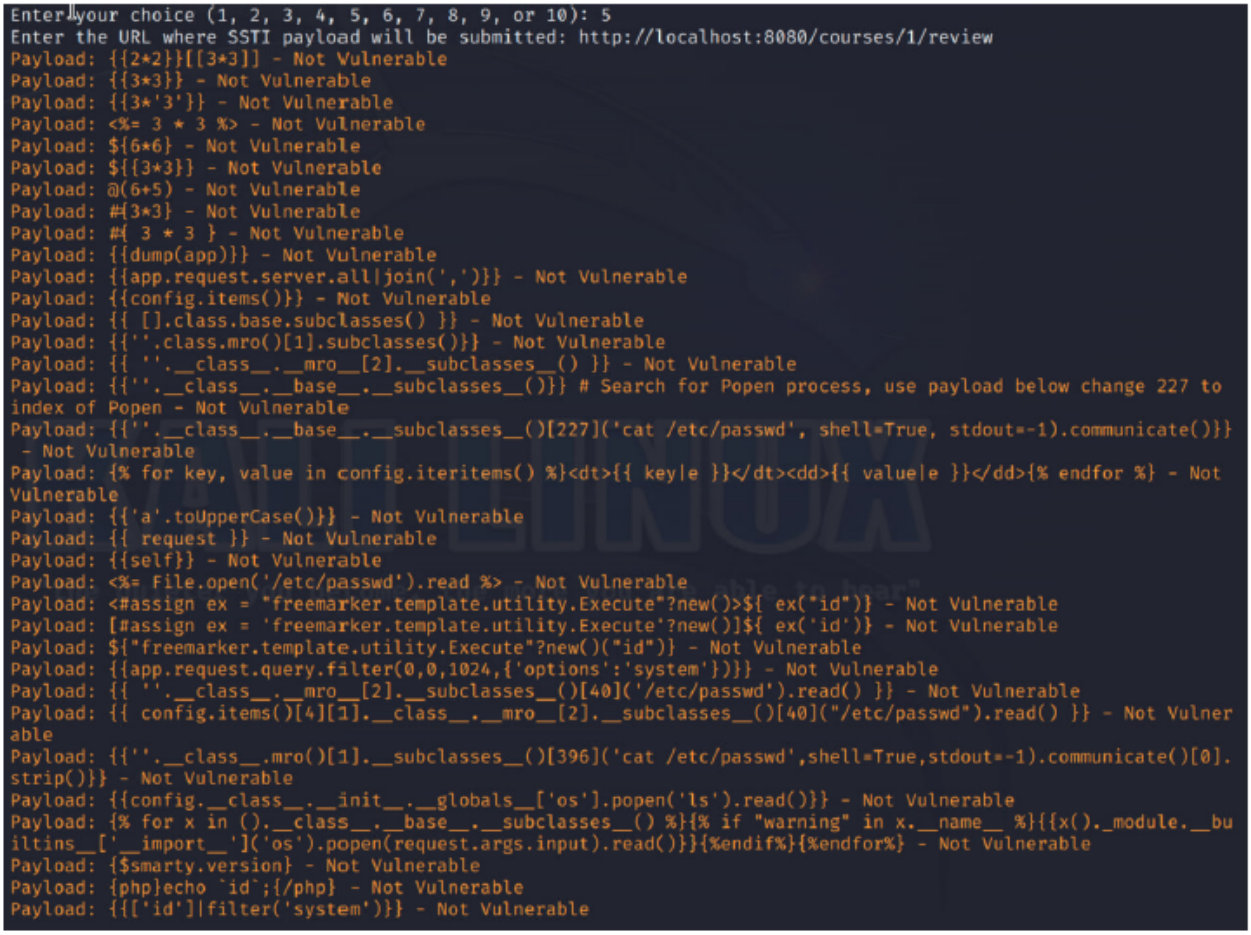

Рисунок 3.8 – Сканування сайту на вразливість до SSTi.

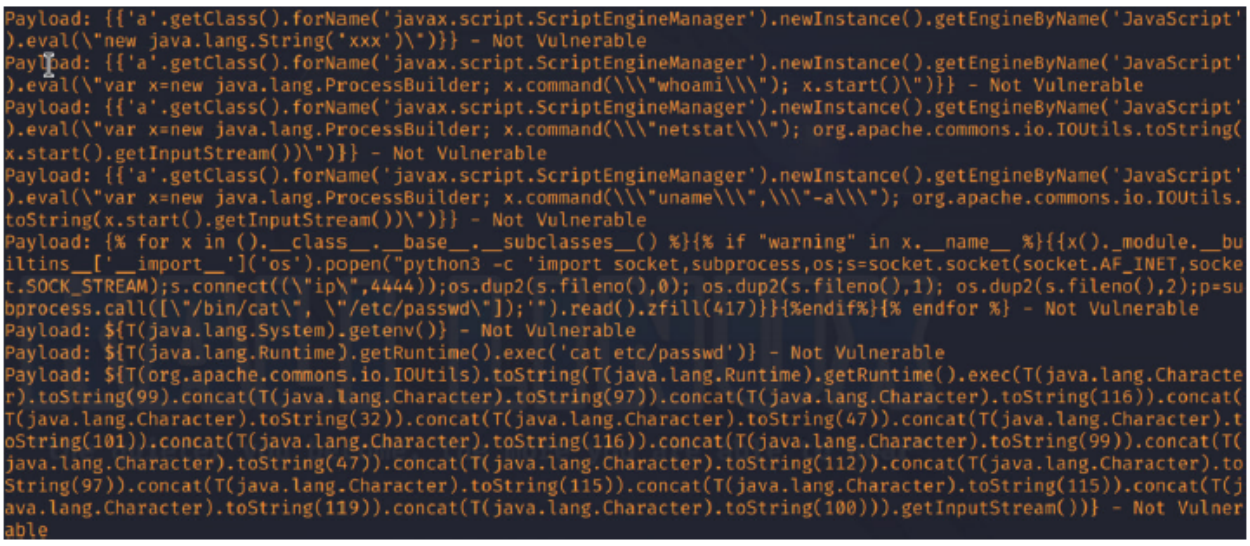

Рисунок 3.9 – Сканування сайту на вразливість до SSTi.

На рисунку 3.10 продемонстровано сканування сайту на вразливість до Clickjacking, на рисунках 3.11 та 3.12 – до Open Redirect.

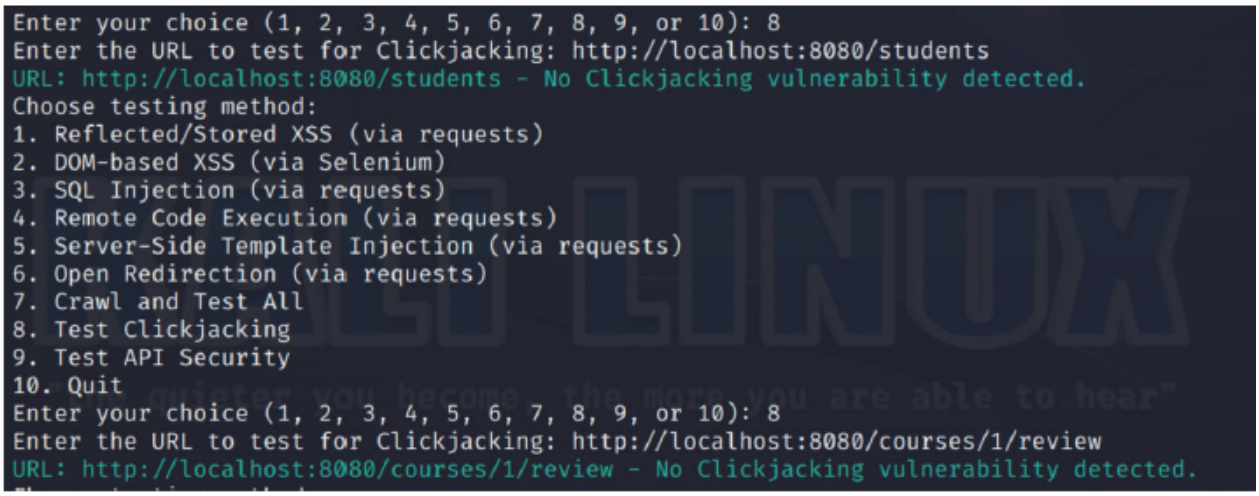

Рисунок 3.10 – Сканування сайту на вразливість до Clickjacking.

Рисунок 3.11 – Сканування сайту на вразливість до Open Redirect.

Рисунок 3.12 – Сканування сайту на вразливість до Open Redirect

На рисунку 3.13 зображена сторінка аутентифікації.

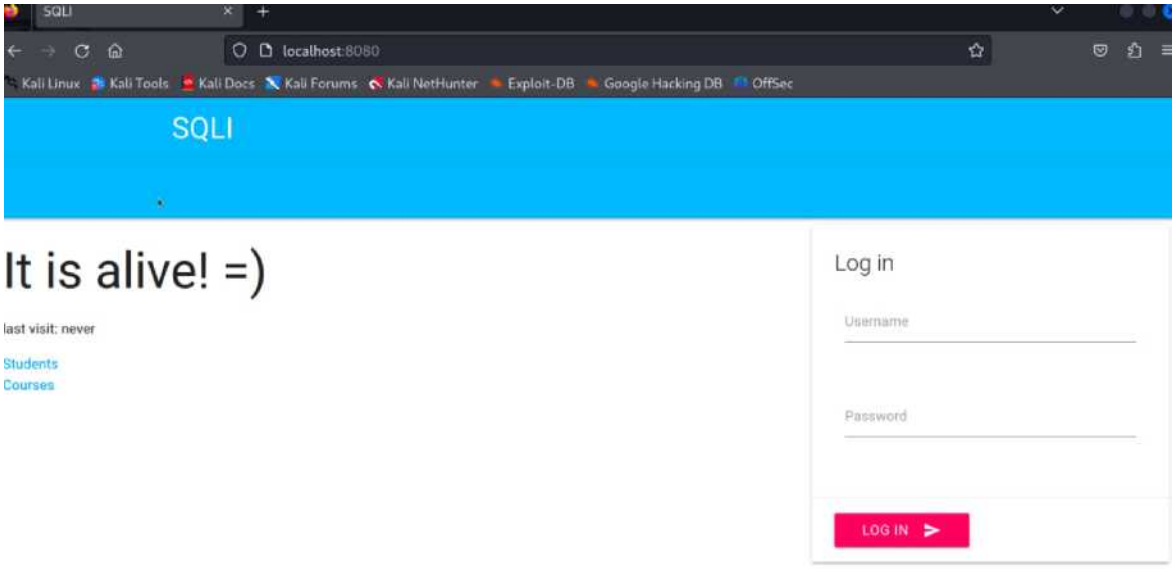

Рисунок 3.13 – Сторінка аутентифікації.

На рисунках 3.14 та 3.15 продемонстровано використання Stored XSS. Додаток не використовує санітізацію та валідацію вводу. Також не використовується autoescape=True в опціях движку шаблонів.

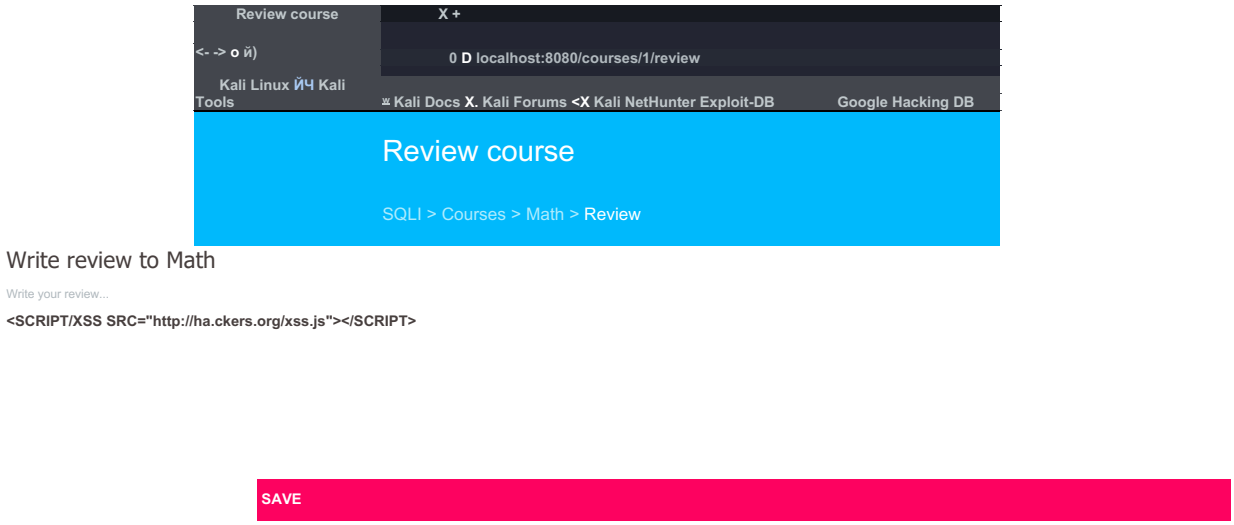

Рисунок 3.14 – Відправка зловмисного коду на сервер через форму.

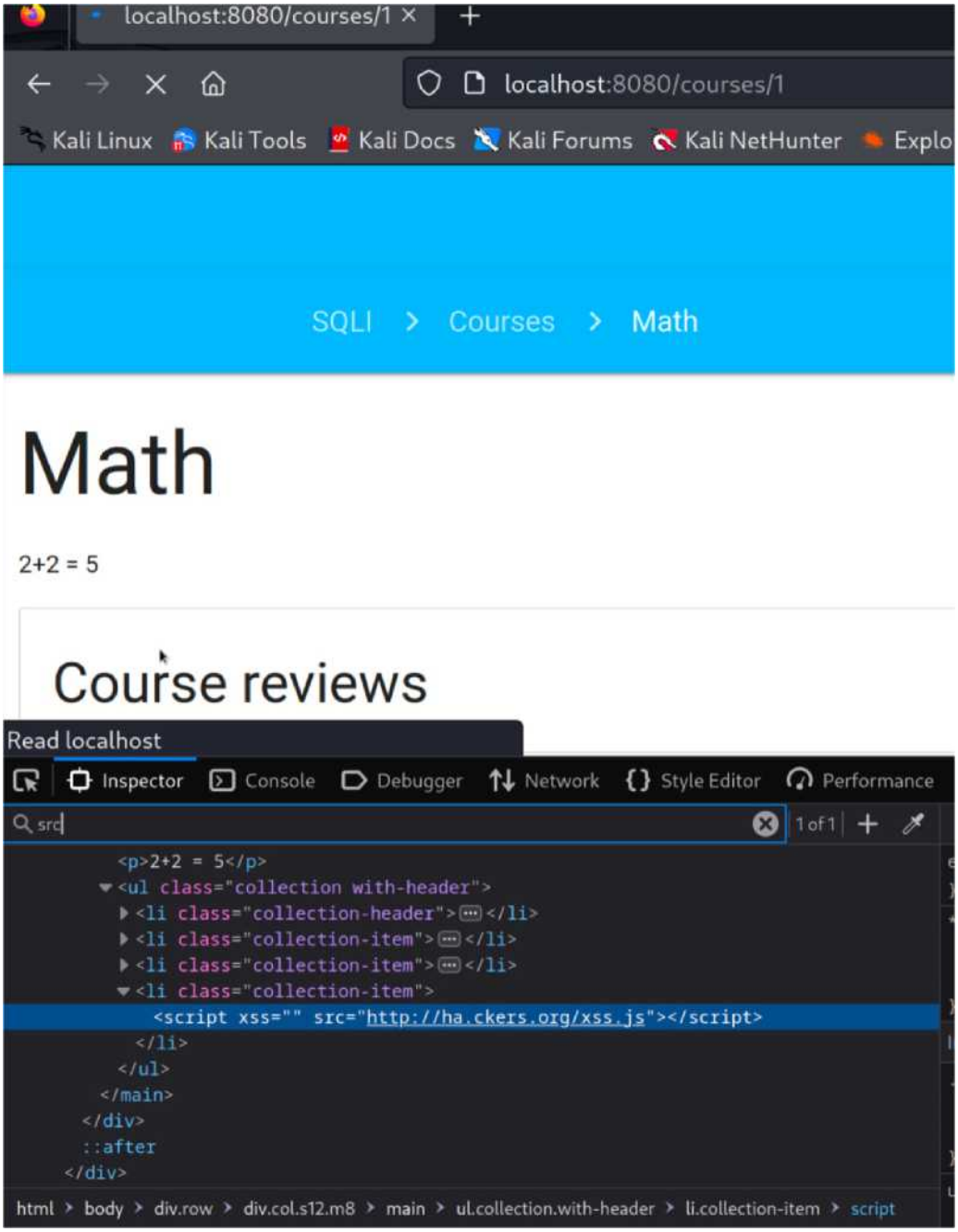

Рисунок 3.15 – Вдала спроба Stored XSS.

Далі на рисунках 3.16 та 3.17 продемонстровано використання SQL Injection. Додаток не використовує санітізацію та валідацію вводу. Замість параметризованих запитів, використовується конкатенація вводу та строки SQL запиту.

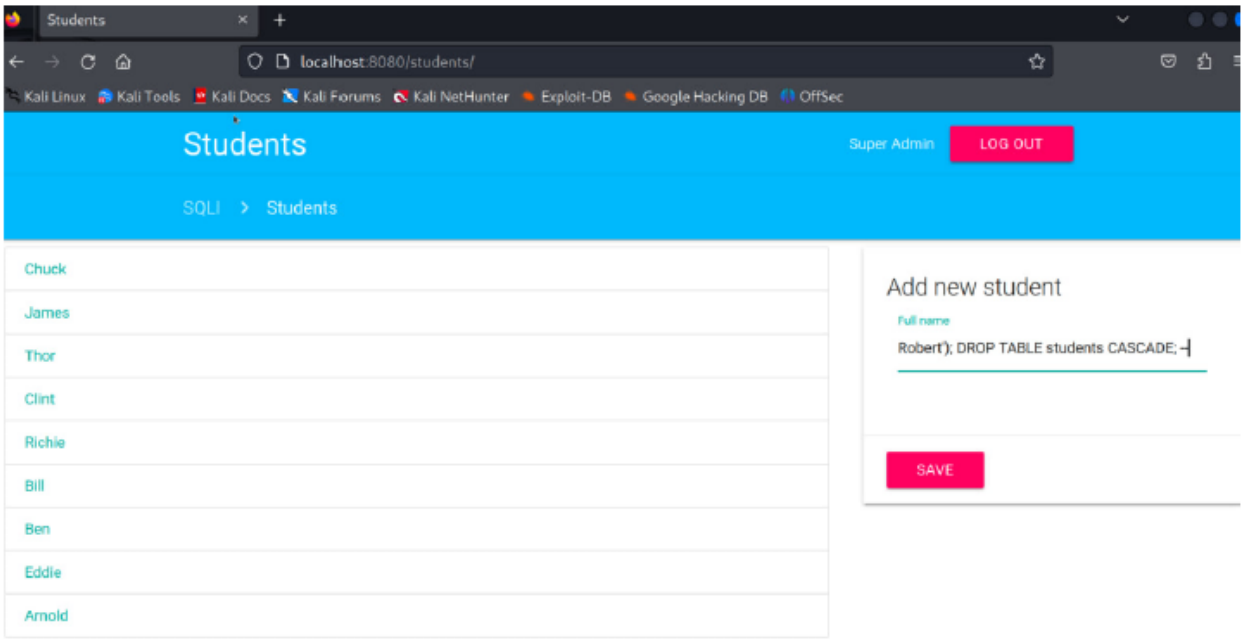

Рисунок 3.16 – Відправка зловмисного SQL виразу на сервер через форму.

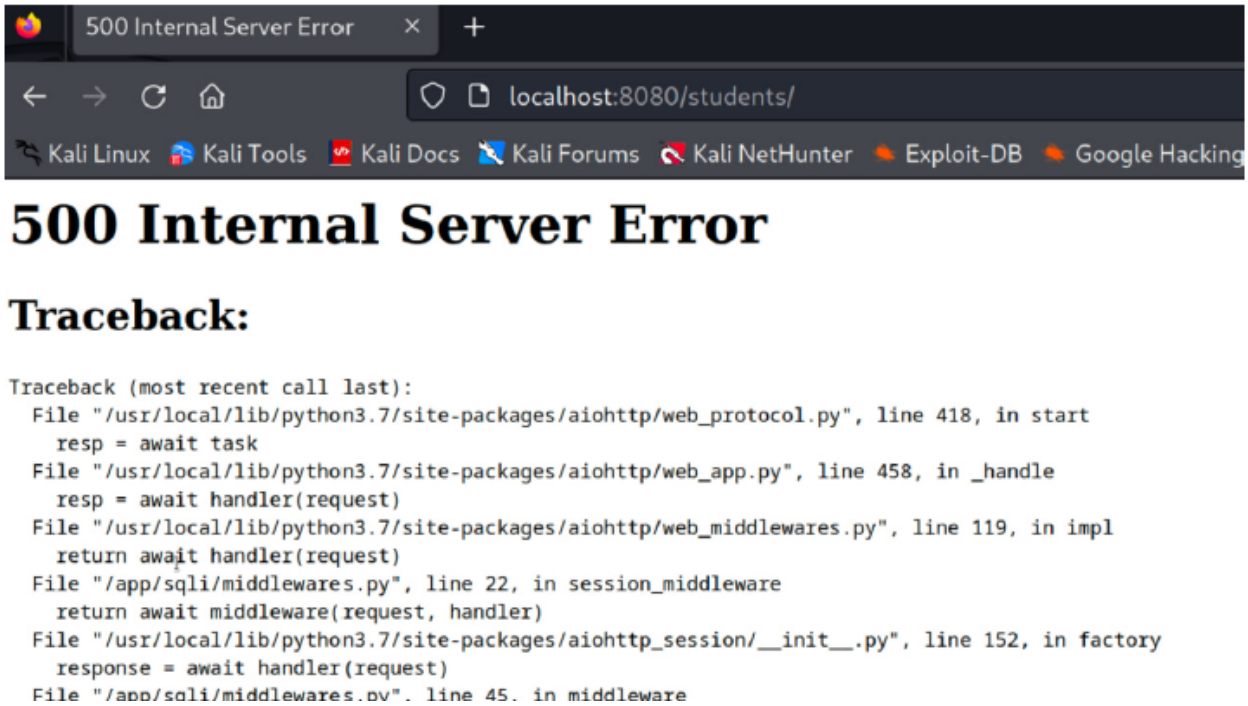

Рисунок 3.17 – Вдала спроба SQLi.

ВИСНОВКИ

У рамках кваліфікаційної роботи був розроблений та протестований простий сканер для виявлення вразливостей вебсайту на прикладі локального навмисно вразливого проекту. Були реалізовані перевірки на вразливість вебдодатку до SQL Injection, Stored/Reflected XSS, DOM Based XSS, Clickjacking, Open Redirect, Remote Code Execution, Server-Side Template Injection та тестування API безпеки. В свою чергу, був підібраний проєкт, який вразливий тільки до SQL Injection, Stored XSS, підробки сесії. В результаті сканер показав правдиві результати.

Це підкреслює здатність сканера ефективно ідентифікувати різні типи вразливостей, що є ключовим для забезпечення загальної безпеки вебдодатків. Важливим аспектом роботи було використання Python та його бібліотек, які надали гнучкість та ефективність у процесі розробки. Адаптація сканера до різних сценаріїв вразливостей, зокрема до специфічних умов локального тестового проєкту, демонструє його потенціал для використання у ширшому діапазоні вебдодатків.

Результати тестування сканера на навмисно вразливому проєкті довели його здатність точно виявляти специфічні вразливості. Це підтверджує важливість ретельного та цілеспрямованого підходу до розробки інструментів безпеки, зокрема, у контексті широкого спектру потенційних кіберзагроз.

У цілому, успішне виконання цієї роботи підкреслює значення розробки простих інструментів для виявлення вразливостей, особливо в умовах постійних кіберзагроз. Сканер, розроблений у цій роботі, не лише демонструє потенціал для подальшого розвитку та удосконалення, але й слугує як цінний інструмент для підвищення рівня безпеки вебдодатків.

Подібне ПЗ є особливо актуальним для тестування простих вебдодатків, які розроблені на застарілому стеку технологій або нашвидкоруч.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Wu, Z., Shen, S., Zhou, H., Li, H., Lu, C., & Zou, D. (2021). An effective approach for the protection of user commodity viewing privacy in e-commerce website. Knowledge-Based Systems, 220, 106952.

2. Aydos, M., Aldan, Q., Co§kun, E., & Soydan, A. (2022). Security testing of web applications: A systematic mapping of the literature. Journal of King Saud University-Computer and Information Sciences, 34(9), 6775-6792.

3. Willberg, M. (2019). Web application security testing with owasp top 10 framework.

4. Devi, R. S., & Kumar, M. M. (2020, June). Testing for security weakness of web applications using ethical hacking. In 2020 4th International Conference on Trends in Electronics and Informatics (ICOEI)(48184) (pp. 354-361). IEEE.

5. Durai, K. N., Subha, R., & Haldorai, A. (2021). A novel method to detect and prevent SQLIA using ontology to cloud web security. Wireless Personal Communications, 117(4), 2995-3014.

6. Kaur, J., Khan, A. I., Abushark, Y. B., Alam, M. M., Khan, S. A., Agrawal, A., ... & Khan, R. A. (2020). Security risk assessment of healthcare web application through adaptive neuro-fuzzy inference system: a design perspective. Risk Management and Healthcare Policy, 355-371.

7. Ansari, M. T. J., Agrawal, A., & Khan, R. A. (2022). DURASec: Durable Security Blueprints for Web-Applications Empowering Digital India Initiative. EAI Endorsed Transactions on Scalable Information Systems, 9(4), e7-e7.

8. Nasereddin, M., ALKhamaiseh, A., Qasaimeh, M., & Al-Qassas, R. (2023). A systematic review of detection and prevention techniques of SQL injection attacks. Information Security Journal: A Global Perspective, 32(4), 252-265.

9. Alghawazi, M., Alghazzawi, D., & Alarifi, S. (2022). Detection of sql injection attack using machine learning techniques: a systematic literature review.

Journal of Cybersecurity and Privacy, 2(4), 764-777.

10. Hasan, M., Balbahaith, Z., & Tarique, M. (2019, November). Detection of SQL injection attacks: a machine learning approach. In 2019 International Conference on Electrical and Computing Technologies and Applications (ICECTA) (pp. 1-6). IEEE.

11. Alenezi, Mamdouh, Muhammad Nadeem, and Raja Asif. «SQL injection attacks countermeasures assessments.» Indonesian Journal of Electrical Engineering and Computer Science 21, no. 2 (2021): 1121-1131.

12. Chen, D., Yan, Q., Wu, C., & Zhao, J. (2021). Sql injection attack detection and prevention techniques using deep learning. In Journal of Physics: Conference Series (Vol. 1757, No. 1, p. 012055). IOP Publishing.

13. Hlaing, Z. C. S. S., & Khaing, M. (2020, February). A detection and prevention technique on sql injection attacks. In 2020 IEEE Conference on Computer Applications (ICCA) (pp. 1-6). IEEE.

14. Ma, L., Zhao, D., Gao, Y., & Zhao, C. (2019, September). Research on SQL injection attack and prevention technology based on web. In 2019 International Conference on Computer Network, Electronic and Automation (ICCNEA) (pp. 176-179). IEEE.

15. Jemal, I., Cheikhrouhou, O., Hamam, H., & Mahfoudhi, A. (2020). Sql injection attack detection and prevention techniques using machine learning. International Journal of Applied Engineering Research, 15(6), 569-580.

16. Crespo-Martinez, I. S., Campazas-Vega, A., Guerrero-Higueras, A. M., Riego-DelCastillo, V., Alvarez-Aparicio, C., & Fernandez-Llamas, C. (2023). SQL injection attack detection in network flow data. Computers & Security, 127, 103093.

17. Nixon, I. K. (2021). Standard penetration test State-of-the-art report. In Penetration Testing, volume 1 (pp. 3-22). Routledge.

18. Liqiang, Z., Weiling, C., Rabcan, J., Davydov, V., & Miroshnichenko, N. (2021). Analysis and comparative studies of software penetration testing methods.

19. Radholm, F., & Abefelt, N. (2020). Ethical Hacking of an IoT-device: Threat Assessment and Penetration Testing: A Survey on Security of a Smart

Refrigerator.

20. Dencheva, L. (2022). Comparative analysis of Static application security testing (SAST) and Dynamic application security testing (DAST) by using open-source web application penetration testing tools (Doctoral dissertation, Dublin, National College of Ireland).

21. McKinnel, D. R., Dargahi, T., Dehghantanha, A., & Choo, K. K. R. (2019). A systematic literature review and meta-analysis on artificial intelligence in penetration testing and vulnerability assessment. Computers & Electrical Engineering, 75, 175-188.

22. Ghanem, M. C., & Chen, T. M. (2019). Reinforcement learning for efficient network penetration testing. Information, 11(1), 6.

23. Hamza, Z. A., & Hammad, M. (2020). Testing Approaches for Web and Mobile Applications: An Overview. International Journal of Computing and Digital Systems, 9(4), 657-665.

24. Rodriguez, G. E., Torres, J. G., Flores, P., & Benavides, D. E. (2020). Cross-site scripting (XSS) attacks and mitigation: A survey. Computer Networks, 166, 106960.

25. Kumar, S., Pathak, S. K., & Singh, J. (2022). A Comprehensive Study of XSS Attack and the Digital Forensic Models to Gather the Evidence. ECS Transactions, 107(1), 7153.

26. Kumar, S., Pathak, S., & Singh, J. (2022). An enhanced digital forensic investigation framework for XSS attack. Journal of Discrete Mathematical Sciences and Cryptography, 25(4), 1009-1018.

27. Liu, M., Zhang, B., Chen, W., & Zhang, X. (2019). A survey of exploitation and detection methods of XSS vulnerabilities. IEEE access, 7, 182004-182016.

28. Adamu, J., Hamzah, R., & Rosli, M. M. (2020). Security issues and framework of electronic medical record: A review. Bulletin of Electrical Engineering and Informatics, 9(2), 565-572.

29. Kaur, J., Garg, U., & Bathla, G. (2023). Detection of cross-site scripting

(XSS) attacks using machine learning techniques: a review. Artificial Intelligence Review, 1-45.

30. Mokbal, F. M. M., Dan, W., Imran, A., Jiuchuan, L., Akhtar, F., & Xiaoxi, W. (2019). MLPXSS: an integrated XSS-based attack detection scheme in web applications using multilayer perceptron technique. IEEE Access, 7, 100567-100580.

31. Liu, Z., Fang, Y., Huang, C., & Xu, Y. (2023). MFXSS: An effective XSS vulnerability detection method in JavaScript based on multi-feature model. Computers & Security, 124, 103015.

32. Liu, Z., Fang, Y., Huang, C., & Han, J. (2022). GraphXSS: an efficient XSS payload detection approach based on graph convolutional network. Computers & Security, 114, 102597.

33. Xiao, F., Yang, Z., Allen, J., Yang, G., Williams, G., & Lee, W. (2022, November). Understanding and Mitigating Remote Code Execution Vulnerabilities in Cross-platform Ecosystem. In Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security (pp. 2975-2988).

34. Ruge, J., Classen, J., Gringoli, F., & Hollick, M. (2020). Frankenstein: Advanced wireless fuzzing to exploit new bluetooth escalation targets. In 29th USENIX Security Symposium (USENIX Security 20) (pp. 19-36).

35. Cai, Z., Wang, A., Zhang, W., Gruffke, M., & Schweppe, H. (2019). 0-days & mitigations: roadways to exploit and secure connected BMW cars. Black Hat USA, 2019(39), 6.

36. Maini, R., BVDUCOEP, P., Pandey, R., Kumar, R., & Gupta, R. (2019). Automated web vulnerability scanner. Int. J. Eng. Appl. Sci. Technol, 4(1), 132-136.

37. Rankothge, W. H., & Randeniya, S. M. (2020, December). Identification and mitigation tool for cross-site request forgery (CSRF). In 2020 IEEE 8th R10 Humanitarian Technology Conference (R10-HTC) (pp. 1-5). IEEE.

38. Sinha, A. K., & Tripathy, S. (2019). CookieArmor: Safeguarding against cross-site request forgery and session hijacking. Security and Privacy, 2(2), e60.

39. Calzavara, S., Conti, M., Focardi, R., Rabitti, A., & Tolomei, G. (2020). Machine learning for web vulnerability detection: the case of cross-site request forgery. IEEE Security & Privacy, 18(3), 8-16.

40. Priyanka, A. K., & Smruthi, S. S. (2020, July). WebApplication Vulnerabilities: Exploitation and Prevention. In 2020 Second International Conference on Inventive Research in Computing Applications (ICIRCA) (pp. 729-734). IEEE.x