Міністерство освіти і науки України
Київський національний університет імені Тараса Шевченка

Факультет інформаційних технологій
Кафедра кібербезпеки та захисту інформації

ДОПУСТИТИ ДО ЗАХИСТУ:
В.о. завідувача кафедри
кібербезпеки
та захисту інформації
_____ Іван ПАРХОМЕНКО
«17» травня 2024 р.

ПОЯСНЮВАЛЬНА ЗАПИСКА

кваліфікаційної роботи

галузь знань *12 Інформаційні технології*
(шифр і назва галузі знань)

спеціальність *125 Кібербезпека*
(код і назва спеціальності)

освітній ступень *магістр*

освітньо-наукова програма *Кібербезпека*
(назва освітньої програми)

на тему: «Модель аналізу веб-експлойтів на основі JavaScript»

Виконавець: студент ІІ курсу, групи КБм-21

_____ **Андрій КУРОЄДОВ**
(підпис) (Ім'я, ПРІЗВИЩЕ)

	Ім'я, ПРІЗВИЩЕ	**Підпис**
Науковий керівник	Сергій БУЧИК	
Нормоконтроль	Інна МИХАЛЬЧУК	

Київ 2024

Міністерство освіти і науки України
Київський національний університет імені Тараса Шевченка

Факультет інформаційних технологій
Кафедра кібербезпеки та захисту інформації

ЗАТВЕРДЖЕНО:
В.о. завідувача кафедри
кібербезпеки
та захисту інформації
_____ Іван ПАРХОМЕНКО
«17» листопада 2023 р.

ЗАВДАННЯ
на виконання кваліфікаційної роботи

спеціальності *125 Кібербезпека*
(код і назва спеціальності)

освітній ступень *магістр*

Здобувача(ки) КБм-21 (група) Куроєдову Андрію Сергійовичу (прізвище ім'я по-батькові)

Тема кваліфікаційної роботи Модель аналізу веб-експлойтів на основі JavaScript

1. ПІДСТАВИ ДЛЯ ПРОВЕДЕННЯ РОБОТИ

Рішення засідання кафедри кібербезпеки та захисту інформації факультету інформаційних технологій протокол № 5 від 15.11.2023 р.

2. МЕТА ТА ВИХІДНІ ДАНІ ДЛЯ ПРОВЕДЕННЯ РОБІТ

Об'єкт досліджень	Процес аналізу вебдодатків на наявність вразливостей.
Предмет досліджень	Вебдодаток, який містить вразливості.
Мета	Розробка моделі аналізу вебексплойтів на основі JavaScript.
Вихідні дані для проведення роботи	Методи аналізу вебексплойтів на основі JavaScript, вебдодаток, який містить вразливості.

3. ОЧІКУВАНІ НАУКОВІ РЕЗУЛЬТАТИ

Наукова новизна Удосконалення моделі аналізу вебексплойтів на основі JavaScript.

Практична цінність Розробка /bin/bash/ скрипта, який дозволяє автоматизувати процес аналізу вебдодатку на наявність вразливостей.

4. ВИМОГИ ДО РЕЗУЛЬТАТІВ ВИКОНАННЯ РОБОТИ

Робота виконана у повному обсязі відповідно до теми.

5. ЕТАПИ ВИКОНАННЯ РОБОТИ

Найменування етапів робіт	Строки виконання робіт (початок-кінець)
Уточнення постановки задачі	17.11.2023 – 18.01.2024
Аналіз літературних джерел	19.01.2024 – 05.02.2024
Ознайомлення з проблематикою аналізу вебдодатків на наявність вразливостей	06.02.2024 – 18.02.2024
Огляд популярних вебвразливостей	19.02.2024 – 24.02.2024
Дослідження вразливих частин коду JavaScript	25.02.2024 – 01.03.2024
Огляд сучасних методологій виявлення вразливостей	02.03.2024 – 12.03.2024
Дослідження наявних заходів та засобів з забезпечення інформаційної безпеки в вебдодатках	13.03.2024 – 18.03.2024
Проєктування моделі аналізу вебдодатків на наявність вразливостей	19.03.2024 – 28.03.2024
Практична реалізація моделі аналізу вебдодатків на наявність вразливостей	29.03.2024 – 10.04.2024
Оформлення пояснювальної записки згідно з методичними рекомендаціями	11.04.2024 – 12.05.2024
Подача пакету документів на розгляд ЕК	13.05.2024 – 18.05.2024

6. РЕАЛІЗАЦІЯ РЕЗУЛЬТАТІВ ТА ЕФЕКТИВНІСТЬ

Економічний ефект	Раннє виявлення вразливостей вебдодатку зменшить витрати на компенсацію коштів для користувачів, які могли постраждати від експлуатації виявлених вразливостей.
Соціальний ефект	Автоматизований аналіз вебдодатку на наявність вразливостей сприяє зменшенню кількості ручної праці розробників та тестувальників, які займаються пошуком вразливостей.

7. ДОДАТКОВІ ВИМОГИ

Завдання видав _____ Сергій БУЧИК
(підпис) (Ім'я, ПРІЗВИЩЕ)

Завдання прийняв до виконання _____ Андрій КУРОЄДОВ
(підпис) (Ім'я, ПРІЗВИЩЕ)

Дата видачі завдання: 17.11.2023 р.
Термін подання кваліфікаційної роботи до ЕК 17.05.2024 р.

РЕФЕРАТ

Пояснювальна записка до кваліфікаційної роботи «Модель аналізу веб-експлойтів на основі JavaScript»: 88 сторінок, 54 рисунків, 50 літературних джерел.

Метою роботи є розробка моделі аналізу вебексплойтів на основі JavaScript.

Для досягнення мети необхідно виконати такі завдання:

- дослідити проблематику аналізу вебексплойтів на основі JavaScript;
- проаналізувати наявні методології аналізу вебдодатків на наявність вразливостей;
- розробити модель аналізу вебексплойтів на основі JavaScript.

Об’єктом дослідження є процес аналізу вебдодатків на наявність вразливостей.

Предметом дослідження є вебдодаток, який містить вразливості.

Практичною цінністю даної роботи є - розробка /bin/bash/ скрипта, який дозволяє автоматизувати процес аналізу вебдодатку на наявність вразливостей.

Наукова новизна полягає в удосконаленні моделі аналізу вебексплойтів на основі JavaScript.

Методи дослідження: аналіз та синтез.

Ключові слова: вразливості, вебдодаток, вебсайт, вебексплойт, аналіз вебдодатків, пошук вразливостей, SQL, XSS, CSRF.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ ТА СКОРОЧЕНЬ

CSP	–	Content Security Policy
CSRF	–	Cross-Site Request Forgery
CSS	–	Cascading Style Sheets
DAST	–	Dynamic application security testing
DDoS	–	Distributed denial-of-service attack
DNS	–	Domain Name System
DOM	–	Document Object Model
FTP	–	File Transfer Protocol
HTML	–	HyperText Markup Language
HTTP	–	HyperText Transfer Protocol
IDS	–	Intrusion Detection System
IPS	–	Intrusion Prevention System
JS	–	JavaScript
QA	–	Quality Assurance
SQL	–	Structured Query Language
URL	–	Uniform Resource Locator
W3C	–	The World Wide Web Consortium
XML	–	Extensible Markup Language
XSS	–	Cross Site Scripting
ОС	–	Операційна система
ПЗ	–	Програмне забезпечення
CSP	–	Content Security Policy
CSRF	–	Cross-Site Request Forgery
CSS	–	Cascading Style Sheets

ЗМІСТ

ВСТУП

В сучасному цифровому світі вебексплойти стали однією з найбільш серйозних загроз для безпеки вебдодатків та вебінфраструктури. Ця проблема виникає з того, що вебексплойти дають можливість зловмисникам використовувати вразливості вебдодатків для виконання шкідливого коду або отримання несанкціонованого доступу до системи. Методи аналізу вебексплойтів на основі JavaScript є надзвичайно важливими, оскільки JavaScript є однією з основних мов програмування для веброзробки, і відповідно, використовується практично на кожній вебсторінці.

JavaScript, як основна мова програмування для веброзробки, проникає практично в кожен аспект інтернет-простору. Від створення динамічного змісту до взаємодії з користувачем, JavaScript є невіддільною складовою будь-якого сучасного вебдодатка. Ця широка поширеність робить JavaScript дуже привабливою мішенню для зловмисників, які шукають можливості використовувати вразливості цієї мови для здійснення атак.

Атаки на основі JavaScript можуть мати різноманітні форми і наслідки. Вони можуть містити в себе впровадження шкідливого коду на сторінках вебсайтів для отримання конфіденційної інформації користувачів, викрадення сесійних файлів, а також підміну вмісту сторінок для поширення фішингових атак або розповсюдження шкідливого програмного забезпечення.

Захист вебдодатків та серверів - це багатогранна задача, що поєднує в собі збереження конфіденційності даних, безперебійну роботу ресурсів та запобігання фінансовим втратам. Нехтування безпекою може призвести до серйозних наслідків, таких як втрата репутації, юридичні проблеми та збитки.

Для розробників та адміністраторів веб інфраструктури постійне вдосконалення захисту є критично важливим завданням. Ось декілька ключових методів:

- аудит вебдодатків: регулярні перевірки дозволяють виявити та усунути вразливості до атак;

- захисні механізми: контроль доступу, валідація даних та захист від XSS-атак мінімізують ризики;
- оновлення: регулярні оновлення вебдодатків та серверів унеможливлюють використання вразливостей зловмисниками;
- безпечні фреймворки: використання фреймворків з вбудованими механізмами захисту (Angular, React, Vue.js) додає додатковий рівень безпеки;
- навчання персоналу: підвищення обізнаності розробників та адміністраторів щодо вебексплойтів - це запорука стійкості вебінфраструктури.

Дослідження методів аналізу вебексплойтів JavaScript є ключовим фактором у забезпеченні безпеки вебдодатків та інфраструктури в цифрову епоху. Ця тема потребує постійного розвитку та вдосконалення для гарантування безпеки в онлайн-середовищі.

Метою роботи є розробка моделі аналізу вебексплойтів на основі JavaScript.

Об’єктом дослідження є процес аналізу вебдодатків на наявність вразливостей.

Предметом дослідження є вебдодаток, який містить вразливості.

Практичною цінністю даної роботи є - розробка /bin/bash/ скрипта, який дозволяє автоматизувати процес аналізу вебдодатку на наявність вразливостей.

Наукова новизна полягає в удосконаленні моделі аналізу вебексплойтів на основі JavaScript.

Методи дослідження: аналіз та синтез.

РОЗДІЛ 1
АНАЛІЗ НАЯВНИХ ВЕБЕКСПЛОЙТІВ

1.1 Класифікація вебексплойтів

Експлойтом називають вид шкідливого програмного забезпечення, яке зловмисники експлуатують у своїх інтересах, зазвичай, у шкідливих інтересах.

Дане шкідливе програмне забезпечення може мати вигляд як виконуваного додатку, невеличкого фрагменту програмного коду або ж набору певних команд. Головне завдання зловмисника - використати наявну вразливість у системі чи програмі та експлуатувати вразливе програмне забезпечення, що призведе до різних наслідків. Таким чином, самостійно експлойт не дасть ніяких шкідливих дій без правильного використання вразливостей [4].

Цілями зловмисника може бути отримання конфіденційної інформації, підвищення привілей у системі, завантаження шкідливого додатка, збій у роботі системи тощо [4].

Об’єктами атаки зловмисників можуть бути операційні системи, різні додатки, апаратна складова тощо. Один зі сценаріїв реалізації атаки це - змушення жертви виконати всі потрібні дії для запуску шкідливого програмного забезпечення, наприклад, перейти за посиланням, завантажити файл і т.д. Шкідливе ПЗ можна надіслати у будь-які месенджери чи електронну пошту, окрім цього можна виконати втручання безпосередньо у систему через мережу [4].

Через те, що експлойти розробляються для певних дій їх слід класифікувати за призначеннями [4]:

- для WEB додатків;
- для ОС;
- для офісних додатків та іншого ПЗ;
- для серверів та їх компонентів;
- для WEB сервісів (WordPress, Drupal, Joomla);

- для апаратної складової.

Вразливості, якими користуються зловмисники, зазвичай з’являються при нехтуванні тестування, порушення безпеки самими розробниками. Інколи, розробник програмного застосунку використовує швидке, але небезпечне рішення, яке призводить до появи різних помилок та збоїв. Зловмисники використовують всі можливі подібні рішення [5].

WEB додаток складається з сервера, програми, додаткового програмного забезпечення, різні служби, бази даних тощо, всі ці компоненти - потенційна ціль для зловмисників [3].

WEB сервер складається з програмного та апаратного забезпечення, які за допомогою протоколів передачі даних (HTTP), а також, інші додаткові протоколи для надсилання відповідей на запити клієнтів, які вони надсилають через мережу Інтернет (рис. 1.1) [1].

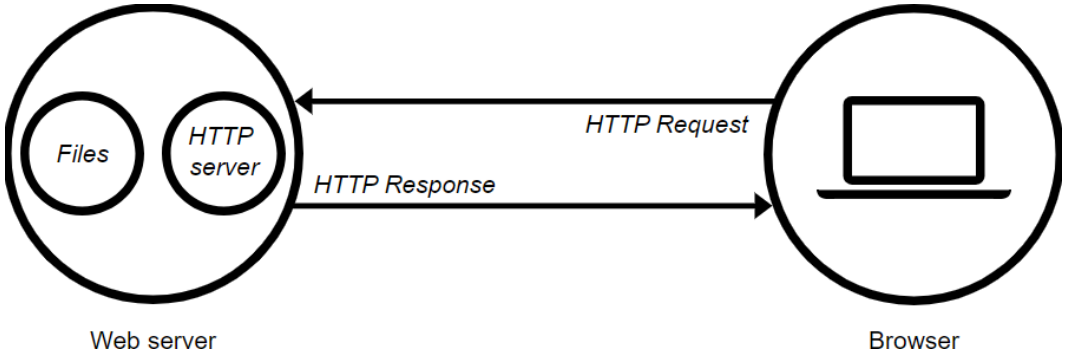


Рисунок 1.1 – Робота вебсервера

Апаратна частина WEB сервера підключена до мережі та забезпечує обмін даних з іншими пристроями в мережі. Щодо програмної складової, вона забезпечує контроль отримання доступу до файлів, які потрібні користувачеві. Роботи вебсерверу - це приклад клієнт-серверної архітектури. Вебсервери використовуються для вебхостингу або розміщення даних для вебсайтів та вебдодатків [2].

Програмне забезпечення серверу поділяється на декілька частин, які контролюють, як користувачі отримують доступ до потрібних файлів. Одна з цих частин - це HTTP-сервер. Сервер HTTP — це сервер, який приймає HTTP запити та

видає HTTP відповіді. HTTP сервери здатні розуміти URL адреси вебсайтів та HTTP протокол (протокол, який дозволяє переглядати веб сторінки для браузерів).

Доступ до вебсервера здійснюється через доменні імена вебдодатків, які забезпечують завантаження потрібного сайту користувачеві, який надіслав запит [2].

Наприклад, коли будь-якому браузеру (Google Chrome, Mozilla Firefox, Opera, Microsoft Edge тощо), потрібен файл, який розміщений на WEB сервері, браузер надсилає запит на доступ до файлу через HTTP. Сервер отримує цей запит, HTTP-сервер прийме запит, знайде вміст і надішле його назад у браузер через HTTP [1].

Докладніше пояснюючи, у процесі отримання вебсторінки, браузер проводить кілька етапів (рис. 1.2). Користувач вводить URL-адресу в адресному рядку браузера, після чого браузер отримує IP-адресу доменного імені через DNS або з кешу. Це дозволяє браузеру зв'язатися з вебсервером. Потім браузер надсилає HTTP-запит для отримання певного файлу від вебсервера. Вебсервер, у свою чергу, надсилає запитану сторінку через HTTP. У випадку відсутності сторінки або виникнення помилки, вебсервер повідомляє про це. Після цього браузер може відобразити вебсторінку. Також важливо відзначити, що на одному вебсервері можуть розміщуватися кілька доменів [1].

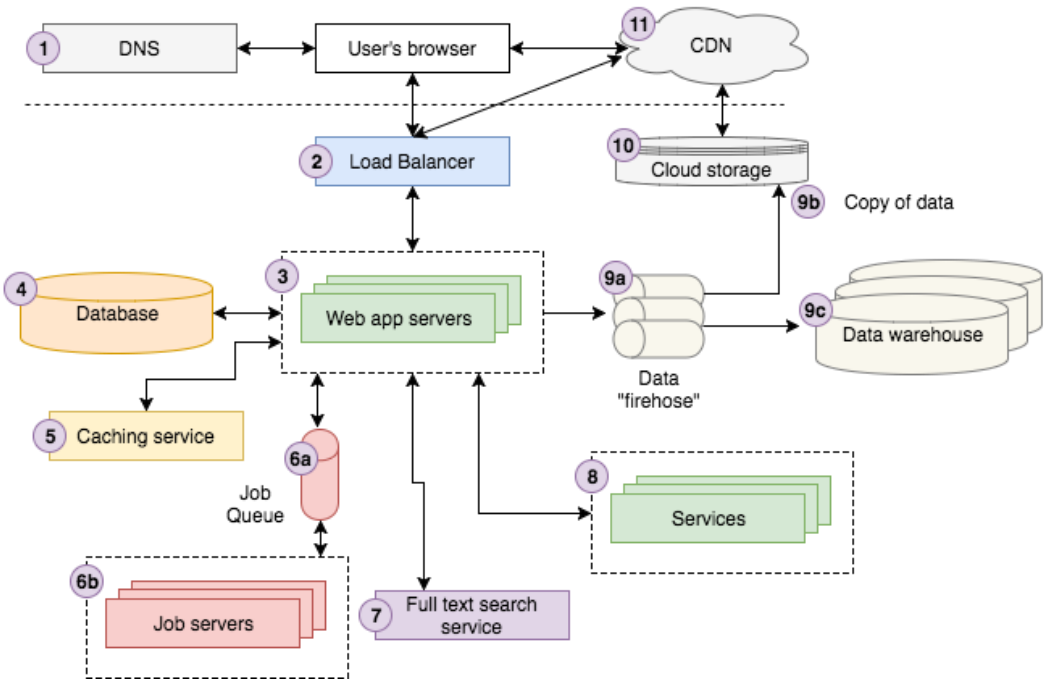


Рисунок 1.2 – Будова сучасної вебархітектури

Атака на WEB сервер та його компоненти може бути максимально простою, наприклад, вебсервер може працювати дуже повільно, коли отримує велику кількість запитів. У суспільстві ця атака відома всім як DDoS атака (відмова в обслуговуванні). Серйозніші атаки можуть містити в собі установку вірусів на сервер або несанкціонований доступ до конфіденційної інформації. Також можливі випадки пошкодження вмісту вебсайту або видалення коду, що можуть бути помічені з великої відстані. Крім того, існує можливість несанкціонованого використання ресурсів сервера для майнінгу криптовалюти, що на сьогодні є популярним способом заробітку [3].

Комунікація між вебклієнтами/браузерами та серверами здійснюється за допомогою різних протоколів, таких як HTTP, HTTPS, FTP та інші. Недоліки у використанні цих протоколів можуть бути використані для здійснення атак. Ці протоколи розташовані на різних рівнях стека протоколів. Хоча вебексплойти здійснюються на рівні додатків (рівень 7), вони можуть впливати на інші рівні через передачу пакетів в ході комунікації (рівень каналу даних) або передачу SYN-пакетів (рівень мережі). Однак атаки на вебсервери на рівні додатків стають більш поширеними, ніж атаки мережевого рівня [3].

1.2 Роль JavaScript у веброзробці

Мова програмування JavaScript відома своєю універсальністю та широким застосуванням у різноманітних виробничих середовищах. Розробники використовують JavaScript для створення динамічного та інтерактивного контенту на вебсайтах і вебдодатках. Ця мова програмування працює безпосередньо у браузері, дозволяючи зробити вебресурс більш цікавим та гарним для користувача та оптимізує роботу в мережі Інтернет [6].

Завдяки JavaScript розробники можуть [6]:

- змінювати елементи HTML;
- керувати об’єктною моделлю документа (DOM);
- взаємодіяти з іншими технологіями, такими як CSS.

Ці можливості дозволяють створювати інтуїтивний інтерфейс користувача, здійснювати перевірку форм та реалізувати динамічне оновлення вмісту [6].

JavaScript стала невід'ємною складовою сучасної веброзробки завдяки своїй гнучкості, простоті використання та сумісності з різними платформами. За результатами опитування розробників Stack Overflow у 2023 році, JavaScript визнана найпопулярнішою мовою програмування серед фахівців вже десятий рік поспіль [6].

Сьогодні у світі існує близько 1,6 мільярда вебсайтів, і понад 95% з них (1,52 мільярда) використовують JavaScript (рис. 1.3). Вона широко використовується для розробки вебдодатків, зокрема у сфері інтерфейсу, який акцентується на естетиці та анімації. Якщо мета розробника це - розробка вебдодатків, які націлені на інтуїтивність та взаємодію з користувачем, тоді важливо використовувати JavaScript. Всі показники свідчать про те, що в найближчому майбутньому JavaScript стане визначальною мовою для створення вебдодатків. Для успішного використання вебпрограм, необхідно розуміти код, що використовується для їх створення [7].

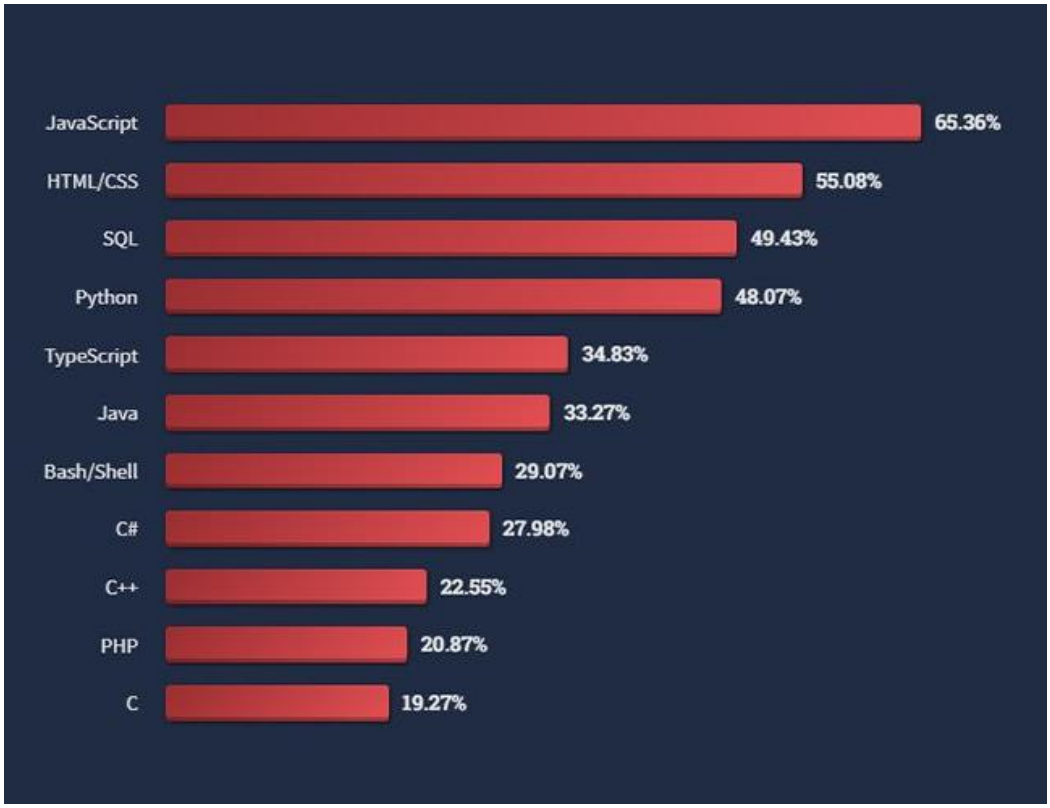


Рисунок 1.3 – Популярність JavaScript

JavaScript за замовчуванням не має вбудованої моделі дозволів безпеки, що робить його чутливим до зловмисних об’єктів [8].

Хоча браузери встановлюють дозволи безпеки відповідно до стандарту W3C, відповідальність за їх керування лежить на власниках вебсайтів [8].

Ця вразливість ще більше підсилюється за умови, що більшість JavaScript, використовуваний у вебпрограмах, походить із бібліотек з відкритим кодом або сторонніх бібліотек, які часто мають свої власні уразливості [8].

Уразливості в JavaScript можуть проникати у вебпрограми різними способами, включаючи помилки в кодуванні, зроблені співробітниками з найкращими намірами [8].

У деяких випадках, вихідний код, який використовується у вебпрограмах, незалежно від того, чи це платформи з відкритим кодом, чи сторонні бібліотеки, може містити недоліки або навіть бути зараженим зловмисним кодом. Крім того, існують більш небезпечні загрози, коли кіберзлочинці зловживають JavaScript, маніпулюючи кодом для отримання доступу до вебпрограм та збору конфіденційних даних [8].

Таким чином, уразливості JavaScript потрапляють у вебпрограми трьома способами [8]:

- внутрішні помилки кодування JavaScript: іноді ненавмисне додавання коду співробітниками може відкрити вразливі місця, що призведе до потенційного витоку даних;
- несправний код через підключення додаткових бібліотек JavaScript: вебпрограми часто містять код із різних джерел, у тому числі сторонніх бібліотек. Будь-який недолік у цьому вихідному коді підвищує ризик кібератаки;
- навмисні зміни коду зловмисниками: кіберзлочинці, маніпулюючи кодом JavaScript, можуть використовувати вебпрограми, викрадаючи конфіденційні дані в процесі.

Використання JavaScript для атак на вебдодатки має як переваги, так і обмеження, які варто врахувати при аналізі вебексплойтів [7].

Переваги використання JavaScript для атак [7]:

- широка поширеність: JavaScript є однією з найпоширеніших мов програмування для веброзробки. Через це, багато вебдодатків містять великий обсяг JavaScript-коду, що створює потенційно вразливі місця для атак;
- вбудовані можливості в браузері: JavaScript виконується безпосередньо в браузері, що дозволяє виконувати скрипти на стороні клієнта і маніпулювати вебсторінками під час їх завантаження або після цього;
- висока інтерактивність: використання JavaScript дозволяє створювати динамічний та інтерактивний контент на вебсторінках, що робить атаки більш ефективними, оскільки користувачі мають тенденцію більше довіряти взаємодії з вебсторінками, які реагують на їхні дії;
- великий вибір інструментів: існує велика кількість інструментів та бібліотек JavaScript, які можуть бути використані для створення атак, від найпростіших до найбільш складних.

Обмеження використання JavaScript для атак [7]:

- обмежені можливості на стороні клієнта: JavaScript виконується на стороні клієнта, що означає, що він має обмежений доступ до локальних ресурсів та функцій браузера, що може ускладнити певні типи атак;
- обмежені права безпеки: Безпека вебдодатків може бути підсилена за допомогою механізмів, таких як Content Security Policy (CSP), які обмежують виконання JavaScript-коду на стороні клієнта та зменшують вразливість вебдодатків до атак;
- відсутність можливості змінювати серверний код: хоча JavaScript може використовуватися для маніпулювання вмістом вебсторінок та взаємодії з користувачем, він не може безпосередньо змінювати серверний код, що обмежує обсяг можливих атак;
- необхідність використання уразливостей в клієнтському коді: для успішних атак на основі JavaScript, зловмисникам часто потрібно використовувати вже наявні вразливості у клієнтському коді вебдодатка, що може виявитися складним завданням в сучасних добре захищених додатках.

Отже, використання JavaScript для атак на вебдодатки має свої переваги, але вимагає уважного аналізу та врахування його обмежень для успішної реалізації атак [7].

1.3 Класифікація атак на вебдодатки

Атаки на вебдодатки - це зловмисна діяльність, спрямована на використання вразливостей у конструкції або реалізації програм, що працюють через веб. Ці атаки можуть призвести до незаконного доступу, крадіжки даних або інших шкідливих наслідків [9].

Найпоширеніші методи атак JavaScript включають в себе виконання шкідливих сценаріїв, викрадення даних сесії користувача або даних з локального сховища вебпереглядача, спонукання користувачів до виконання непередбачуваних дій та експлуатацію вразливостей у вихідному коді вебпрограм (рис. 1.4) [9].

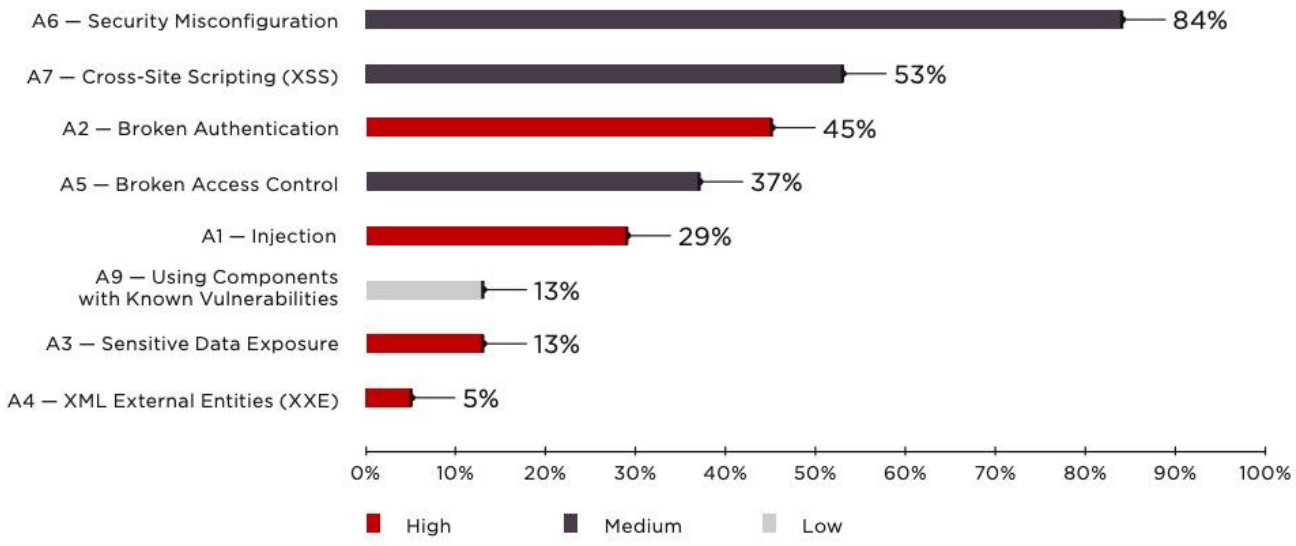


Рисунок 1.4 – Найпоширеніші типи вебатак

Найпоширеніші вебвразливості та відповідні заходи щодо їх запобігання них:

1. SQL-ін’єкція: результат прийняття ненадійного введення без належної перевірки. Приклади включають впровадження SQL, LDAP та заголовка HTTP.

Зловмисники можуть вставити шкідливий SQL-код у поля введення вебдодатків, що може призвести до несанкціонованого доступу до бази даних. WAF може запобігти атакам SQL-ін’єкцій, блокуючи будь-які підозрілі SQL-запити та використовуючи методи зіставлення шаблонів для виявлення та блокування спроб SQL-ін’єкцій [3].

2. Неправильна конфігурація: виникає, коли налаштування не зберігаються належним чином під час ручного виконання процесів [3].

3. Міжсайтовий сценарій (XSS): виникає, коли сервер приймає ненадійний код JavaScript через введення користувача, а потім повертає його у відповідь, що призводить до виконання цього коду в браузері [10].

Міжсайтовий сценарій (XSS) представляє собою вид атаки на вебдодатки, що полягає у введенні шкідливих сценаріїв на вебсторінки, які переглядають інші користувачі. Це зазвичай відбувається шляхом введення сценарію у поля введення форми або параметр URL-адреси, який потім зберігається у базі даних вебдодатку (рис. 1.5) [7].

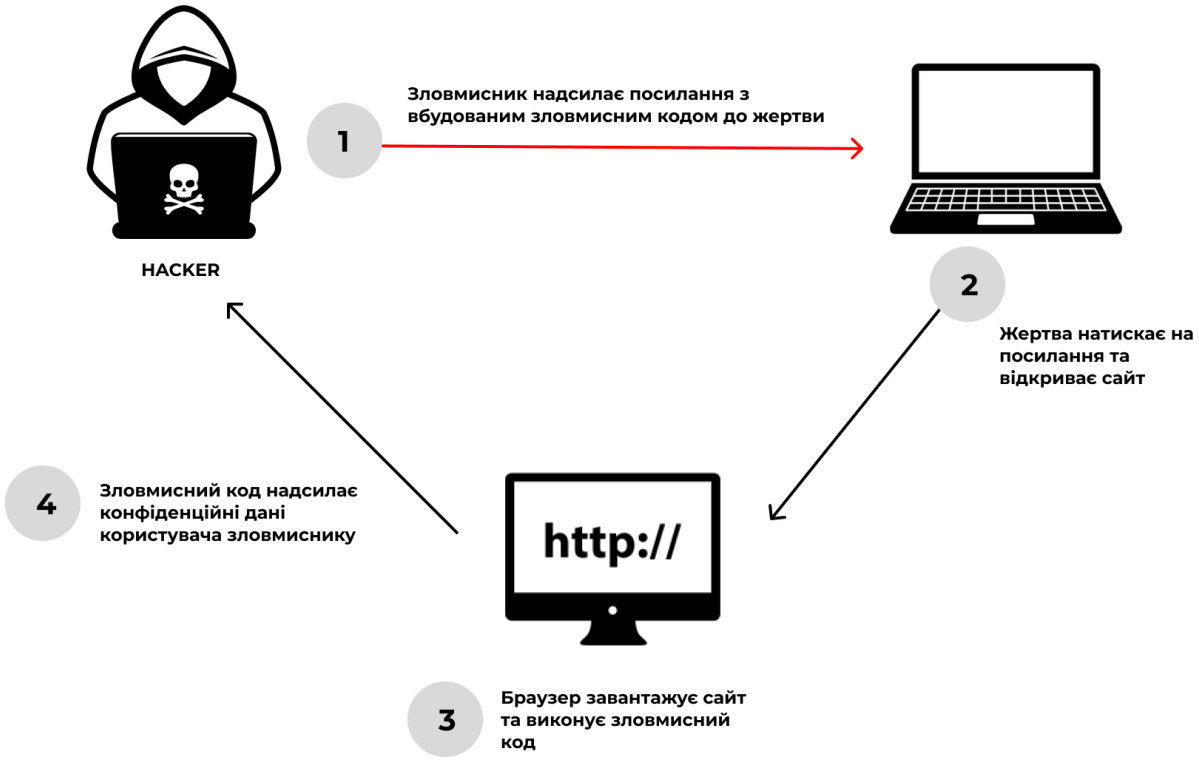


Рисунок 1.5 – Типові кроки в експлойті XSS

Коли інший користувач відвідує сторінку зі шкідливим сценарієм, цей сценарій виконується у його браузері, що може дозволити зловмиснику викрасти дані або здійснити інші шкідливі дії від імені користувача. Щоб запобігти атакам XSS, необхідно належним чином очищати дані, що вводяться користувачем, використовувати заголовки політики безпеки вмісту (CSP) та усувати ненадійні дані [11].

4. Міжсайтова підробка запитів (CSRF) - це тип атаки на вебпрограму, яка шахрайським шляхом змушує користувача виконати небажану дію з вебпрограмою, в якій він вже автентифікований. Це часто досягається за допомогою надсилання спеціально створеного посилання або сценарію користувачеві, який потім виконує небажану дію після натискання (рис. 1.6) [11].

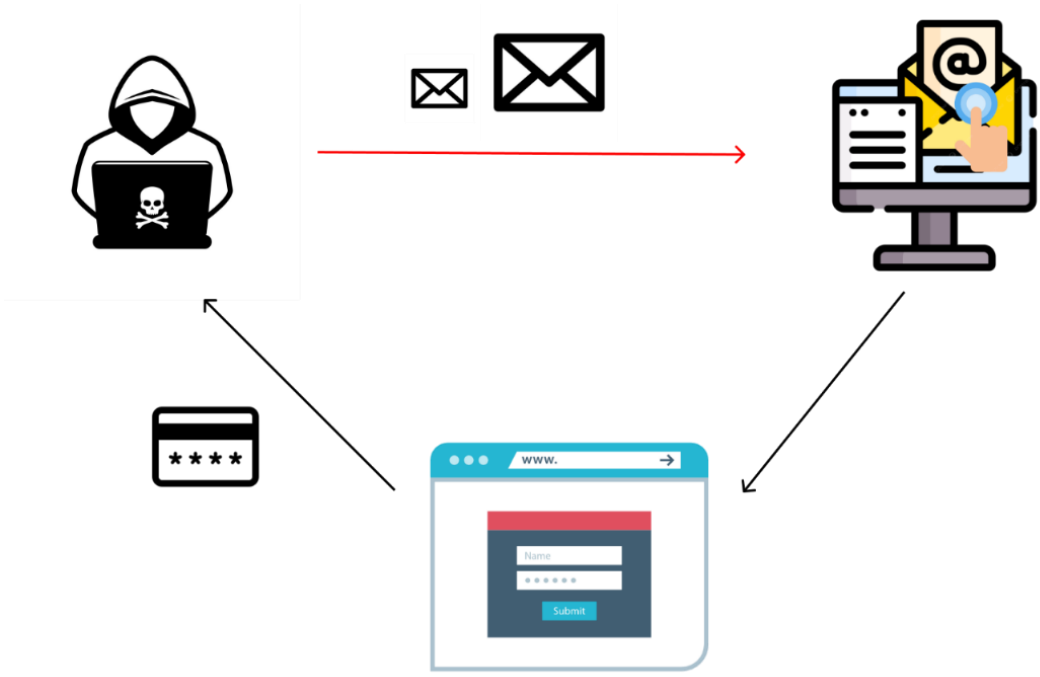


Рисунок 1.6 – Реалізація CSRF

Наприклад, атака CSRF може бути використана для проведення несанкціонованих операцій, таких як покупки або зміни налаштувань облікового запису. Щоб запобігти атакам CSRF, можна використовувати анти-CSRF маркери, що є унікальними ідентифікаторами, які генеруються вебпрограмою для кожного сеансу користувача і мають включатися в кожний запит до програми [11].

5. Застаріле програмне забезпечення - з огляду на зростання використання пакетів програмного забезпечення з відкритим кодом та сторонніх програм, важливо постійно оновлювати їх. Використання застарілого програмного забезпечення може стати загрозою, особливо якщо вразливості стають загальнодоступними [11].

6. HTTP request smuggling - використовує невідповідність у аналізі невідповідних HTTP-запитів, яка виникає при обміні їх між двома HTTP-пристроями, зазвичай між сервером і брандмауером, що підтримує HTTP, або зовнішнім проксі-сервером. Процес транспортування HTTP-запитів відбувається шляхом створення кількох налаштованих HTTP-запитів, які дозволяють двом цільовим сутностям бачити дві різні серії запитів [12].

Заголовок HTTP пропонує два різні способи вказати, де закінчується запит: заголовок Transfer-Encoding і заголовок Content-Length. Уразливість транспортування HTTP-запитів виникає, коли зловмисник надсилає обидва заголовки в одному запиті. Це може призвести до того, що зовнішній або внутрішній сервер неправильно інтерпретує запит через зловмисний HTTP-запит (рис. 1.7) [12].

Уразливості транспортування запитів використовуються кіберзлочинцями для обходу заходів безпеки, отримання доступу до конфіденційної інформації та прямого скомпрометування користувачів різних програм. Також можна використовувати цю уразливість для вторинних експлойтів, таких як обхід брандмауерів, часткове отруєння кешу та міжсайтовий сценарій (XSS) [12].

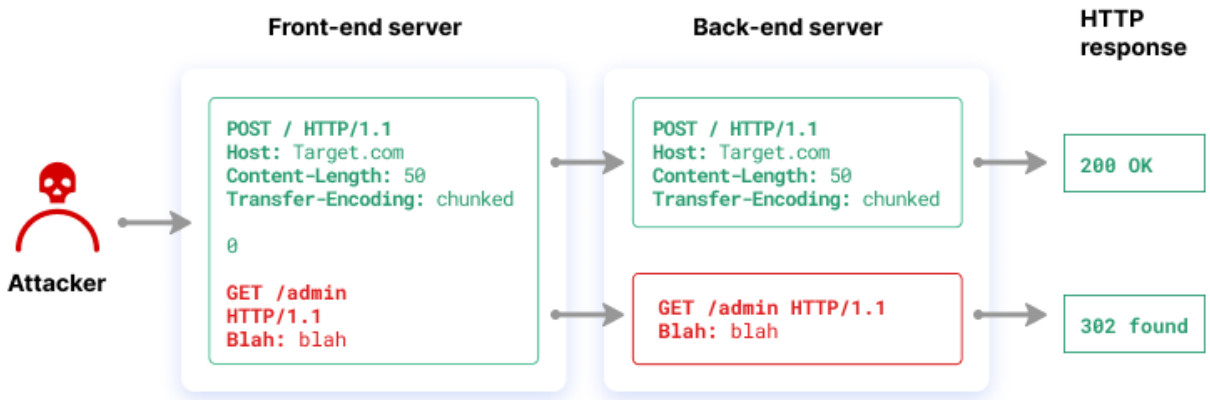


Рисунок 1.7 – Реалізація HTTP request smuggling

7. DDoS (відмова в обслуговуванні) - це спосіб атаки на вебпрограму, що полягає в перевантаженні програми великою кількістю трафіку з різних джерел, таких як ботнети або скомпрометовані пристрої. Це може призвести до того, що вебпрограма стане недоступною для легітимних користувачів (рис. 1.8) [10].

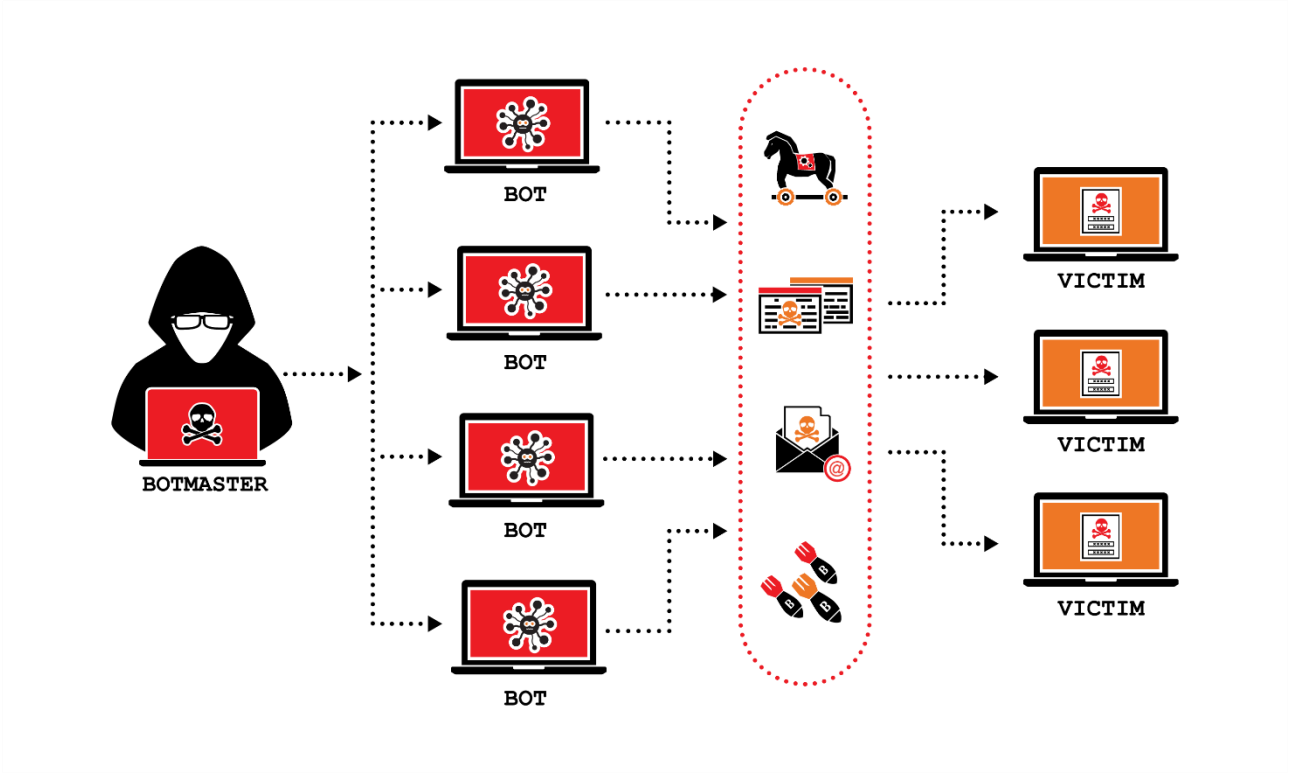


Рисунок 1.8 – Реалізація DDoS атаки

DDoS-атаки можна уникнути за допомогою пристроїв мережевої безпеки, таких як брандмауери та системи запобігання вторгненням, які можуть виявляти та блокувати шкідливий трафік. Крім того, розробники вебдодатків можуть скористатися мережами доставки контенту (CDN) та балансувальниками навантаження для розподілу трафіку між декількома серверами з метою пом’якшення наслідків DDoS-атак [10].

8. Атака XML (XXE) - це форма атаки на вебдодатки, що використовує вразливості у парсерах XML, що використовуються у додатку. Це може дозволити зловмиснику отримати доступ до конфіденційних даних або виконати несанкціоновані дії на сервері вебдодатку (рис. 1.9) [10].

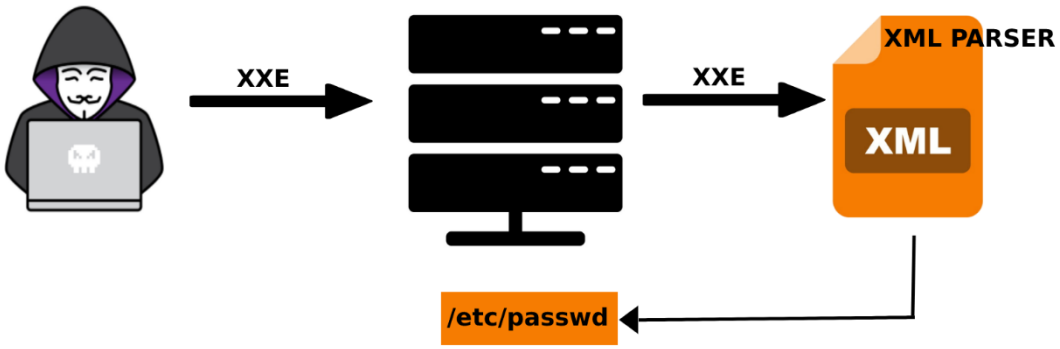


Рисунок 1.9 – Реалізація XXE атаки

Часто атаки XXE включають впровадження спеціально створених корисних даних XML, які використовують можливість синтаксичного аналізатора XML читати зовнішні сутності. Цим атакам можна запобігти, вимкнувши синтаксичний аналіз зовнішніх об’єктів або використовуючи захищені аналізатори XML, які належним чином очищають вхідні дані [10].

9. Автентифікація та авторизація - ідентифікатор сесії може бути розкритим через URL-адресу. Пароль може бути незашифрованим. Якщо тайм-аут реалізовано неправильно, можливе викрадення сесії. Доступ до неавторизованих ресурсів можливий, навіть якщо інтерфейс користувача не розкриває їх [3].

10. Прямі посилання на об’єкти - через поганий дизайн або помилку кодування прямі посилання можуть бути доступні для клієнтів. Наприклад, запит GET на download.php?file=secret.txt може обійти авторизацію та дозволити пряме завантаження захищеного файлу. Іншим прикладом є пряме скидання пароля адміністратора [3].

11. Brute Force (атака грубою силою) - це автоматичний метод вгадування комбінації імені користувача та пароля для отримання несанкціонованого доступу до вебпрограми. Зловмисники використовують програмні інструменти, щоб спробувати різні комбінації імен користувачів і паролів, поки не отримають доступ (рис. 1.10) [10].

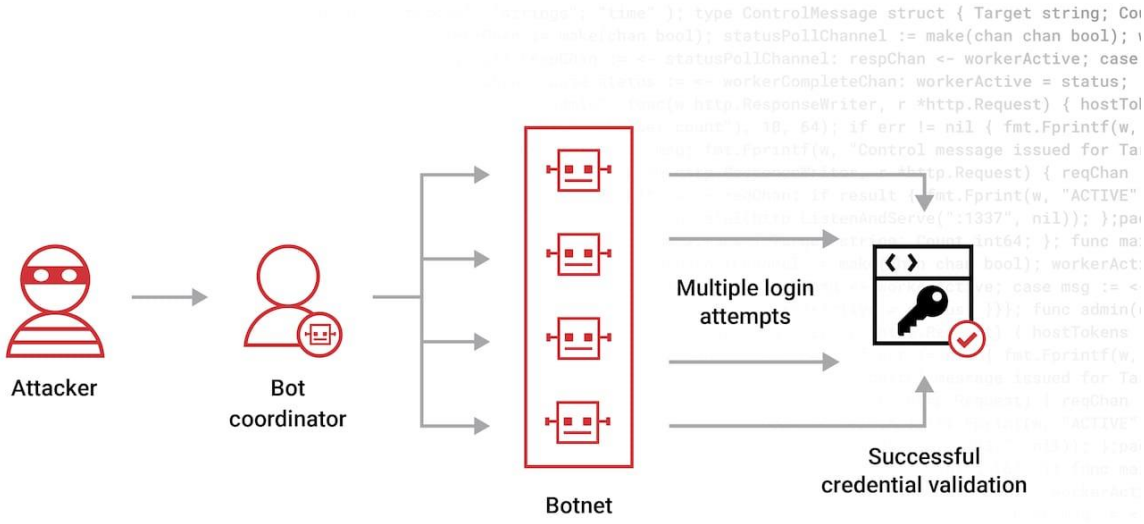


Рисунок 1.10 – Реалізація атаки «Грубої сили (Brute Force)»

Щоб убезпечити себе від атак грубої сили, вебдодатки можуть впроваджувати стратегії контролю швидкості та блокування облікових записів. Контроль швидкості обмежує кількість спроб входу з однієї IP-адреси, тоді як блокування облікового запису тимчасово призупиняє доступ до облікового запису після певної кількості невдалих спроб входу [10].

12. Розкриття даних - одним з ризиків є витік конфіденційних даних, які можуть зберігатися у незашифрованому вигляді або бути відкритими у файлах cookie чи URL-адресах під час взаємодії між клієнтом та сервером без використання HTTPS [3].

Висновки за розділом 1

В першому розділі дипломної роботи було проведено аналіз класифікації вебексплойтів, ролі JavaScript у веброзробці та класифікації атак на вебдодатки. Під час дослідження було виявлено, що класифікація вебексплойтів дозволяє систематизувати та узагальнити різноманітні типи атак, спрямованих на вебдодатки, що відкриває можливості для розробки ефективних методів захисту. Роль JavaScript у веброзробці була визначена як критична, оскільки ця мова програмування

використовується для створення динамічного та інтерактивного контенту на вебсайтах, але водночас може стати об’єктом атак через свою широку поширеність та можливість виконання на стороні клієнта. Класифікація атак на вебдодатки вказує на різноманітність загроз, що можуть виникнути у зв’язку з використанням вебтехнологій, та вимагає відповідного уваги та заходів забезпечення безпеки для запобігання можливим наслідкам таких атак. Результати аналізу першого розділу підкреслюють важливість подальшого вивчення та розробки методів захисту вебдодатків з урахуванням специфіки їхньої розробки та можливих загроз безпеці.

РОЗДІЛ 2
МЕТОДИ АНАЛІЗУ ВЕБЕКСПЛОЙТІВ НА ОСНОВІ JAVASCRIPT

2.1 Загальна концепція та основні принципи аналізу вебексплойтів

Загальна концепція аналізу вебексплойтів включає в себе ряд етапів та принципів, які важливо враховувати для ефективного виявлення, вивчення та захисту від потенційних загроз. Розглянемо ці етапи більш детально [7]:

1. Виявлення вразливостей:

- аналіз вихідного коду: дослідження вихідного коду вебдодатка для виявлення можливих вразливостей, таких як некоректна обробка даних, використання застарілих бібліотек або вразливостей, пов’язаних зі специфічними функціями мов програмування;
- аудит безпеки: систематичний аналіз вебдодатка для виявлення потенційних вразливостей, який може включати в себе тестування введених даних, аналіз структури бази даних, перевірку конфігурації сервера та інші аспекти безпеки.

2. Експлуатація вразливостей:

- використання технік атаки: використання знайдених вразливостей для введення шкідливого коду, викрадення конфіденційної інформації або отримання несанкціонованого доступу до системи;
- ретроспективний аналіз: оцінка ефективності атаки та її потенційного впливу на вебдодаток або вебсервер.

3. Аналіз відповіді на атаку:

- моніторинг системи: відстеження та аналіз реакції системи на атаку, включаючи перевірку журналів подій, виявлення спроб несанкціонованого доступу та інші індикатори компрометації;
- вивчення методів атаки: розуміння та вивчення методів, які використовувалися в атаках, для розробки ефективних захисних стратегій у майбутньому.

4. Попередження та захист [7]:

• впровадження патчів та оновлень: виправлення виявлених вразливостей через впровадження патчів та оновлень програмного забезпечення;

• конфігураційні зміни: налаштування конфігурації сервера та вебдодатка для зменшення ризиків безпеки;

• використання додаткових захисних механізмів: встановлення та налаштування додаткових захисних механізмів, таких як фаєрволи, системи виявлення вторгнень (IDS/IPS) та системи контролю безпеки вмісту (CSP).

Загальна концепція аналізу вебексплойтів полягає в поєднанні ретельного дослідження вразливостей, експлуатації їх для виявлення потенційних загроз, аналізу реакції системи на атаку та розробки заходів для попередження майбутніх інцидентів та захисту від них.

Дослідження вебексплойтів можна реалізувати за допомогою статичного та динамічного аналізів.

Статичний аналіз - це процес оцінки вихідного коду до його виконання (рис. 2.1). У мовах програмування, що компілюються, такий аналіз може виконуватися безпосередньо компілятором. Проте, у мовах з динамічною інтерпретацією, як, наприклад, JavaScript, інструменти статичного аналізу потрібно налаштовувати для оцінки коду за певний час до його виконання [13].

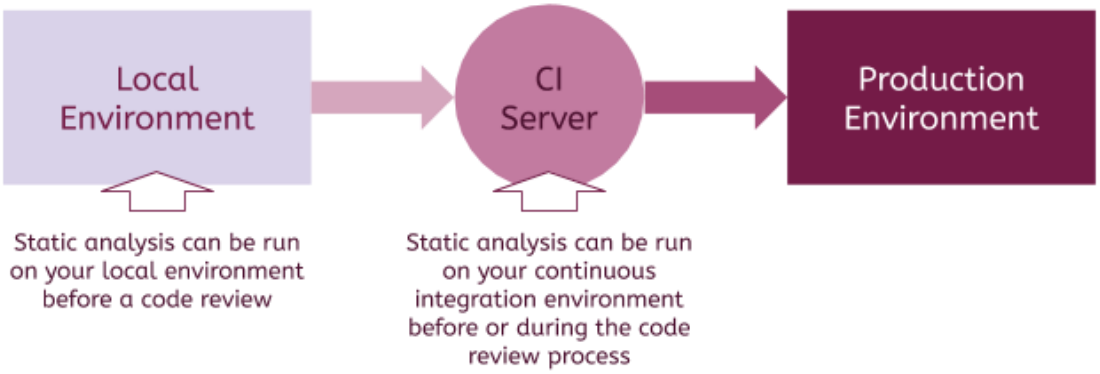


Рисунок 2.1 – Статичний аналіз

Статичні аналізатори виконують глибокий аналіз вашого коду, перетворюючи його у структуру, відому як абстрактне синтаксичне дерево. Ця структура подається на перевірку відповідно до набору правил, встановлених самим аналізатором. В більшості випадків статичні аналізатори також дозволяють розробникам встановлювати власні правила, але це може залежати від конкретного інструменту [14].

Зазвичай статичний аналіз використовується для наступних цілей [14]:

- забезпечення послідовного стилю та форматування коду;
- виявлення типових та потенційних помилок;
- обмеження складності коду;
- перевірка узгодженості типів даних;
- мінімізація ризиків безпеки;
- підтримка актуальності залежностей від сторонніх бібліотек.

При впровадженні статичного аналізу в робочий процес команда зазвичай встановлює та налаштовує необхідні пакети, а потім автоматично запускає аналіз на своєму сервері безперервної інтеграції, перш ніж код буде розгорнуто у робочому середовищі [13].

Крім того, якщо всі члени команди не зобов’язалися використовувати статичний аналіз, можна встановити ці інструменти у своєму локальному середовищі та запустити їх локально перед відправкою коду на перевірку [13].

Як розробники програмного забезпечення, ми маємо багато інструментів, які допомагають підвищити якість коду, включаючи модульні та інтеграційні тести, перевірки коду, ручну перевірку якості та статичний аналіз. Усі ці інструменти корисні, але більшість з них потребує ручного введення від розробника чи тестувальника. Статичний аналіз є винятком [13].

Оскільки статичний аналіз можна повністю автоматизувати, це один із найкращих способів покращити якість коду JavaScript, заощаджуючи час розробника [13].

Незалежно від того, коли і як відбувається статичний аналіз, його мета залишається постійною: допомагати зробити код більш узгодженим, легше

обслуговувати та правильним. Хоча він не замінює автоматизоване чи ручне тестування, статичний аналіз може виявляти помилки, які можуть залишитися непоміченими іншими інструментами забезпечення якості [14].

Статичні аналізатори коду можуть сканувати всю кодову базу на предмет помилок у введенні, виведенні або обробці даних, тоді як динамічні аналізатори перевіряють лише ту частину коду, яка виконується під час виконання програми. Перевага динамічного аналізу полягає у тому, що він допомагає виявити вразливості або загрози, які можуть бути надто складними для статичного аналізу, наприклад, витоки пам’яті, посилання на нульовий вказівник або проблеми з паралельністю [15].

Динамічний аналіз коду оцінює поведінку запущеної програми, спрямований на виявлення можливих ризиків та помилок у програмі. Цей метод використовується для пошуку та виправлення помилок, виявлення вузьких місць у продуктивності або виявлення проблем безпеки [15].

Основна мета динамічного аналізу полягає у виявленні потенційних помилок або загроз безпеці на ранній стадії виконання коду, щоб їх можна було виправити до впровадження продукту у виробництво. Налагодження цих помилок та загроз ще до релізу допомагає уникнути можливих проблем, таких як падіння бізнесу, втрата грошей або клієнтів [15].

Головна відмінність між статичним і динамічним аналізом коду полягає у часі їх виконання. Статичний аналіз коду проводиться перед запуском коду, тоді як динамічний аналіз відбувається під час його виконання. Окрім цього, існують інші різниці між цими двома підходами [15].

Сканери вразливостей вебдодатків, це автоматизовані програмні засоби, які проводять аналіз вебдодатків, зазвичай ззовні, з метою виявлення потенційних проблем безпеки, таких як міжсайтовий сценарій, SQL-ін’єкція, впровадження команд, обхід шляху та незахищена конфігурація сервера. Ці інструменти часто відомі як інструменти динамічного тестування безпеки додатків (DAST). На ринку представлено значну кількість комерційних і відкритих сканерів, кожен з яких має свої переваги та недоліки. Для оцінки ефективності інструментів DAST

рекомендується звернутися до проекту OWASP Benchmark, який науково оцінює продуктивність різних типів сканерів вразливостей, включаючи DAST [16].

Наведено перелік інструментів, які можна використовувати для інтеграції динамічного аналізу коду у програмне забезпечення. Ці інструменти є універсальними і підходять для будь-якого стеку технологій чи мови програмування. Сховище з відкритим вихідним кодом містить список таких інструментів, які ви можете розглянути та вибрати залежно від вашої поточної потреби [15].

2.1.1 Інструменти статичного аналізу вебсторінки

Найпоширенішими інструментами для статичного аналізу є [17, 20]:

DeepSource є одним з найкращих інструментів статичного аналізу коду. Він виявляє понад 170 проблем, пов’язаних з якістю коду, таких як помилки продуктивності, проблеми безпеки та антипатерни. DeepSource підтримує проекти Java на Gradle, а незабаром також буде підтримувати Maven та Android.

Checkstyle - це інструмент статичного аналізу програмного коду, призначений для контролю та забезпечення відповідності стандартам написання коду. Він виявляє порушення встановлених правил, надає рекомендації з їх усунення та автоматично впроваджує виправлення та переформатування коду. Checkstyle легко інтегрується з популярними середовищами розробки, такими як Eclipse, IntelliJ IDEA та NetBeans.

MobSF відомий як Mobile Security Framework (MobSF), є автоматизованим універсальним мобільним додатком для перетестування безпеки та аналізу зловмисного програмного забезпечення. MobSF підтримує різні формати мобільних додатків та архівів, що дозволяє проводити як статичний, так і динамічний аналіз програм (рис. 2.2) [15].

Додаток проводить статичний аналіз додатків на базі ОС Android чи IOS. Дозволяє дослідити безпеку додатка, що сканується та пропонує методи покращення.

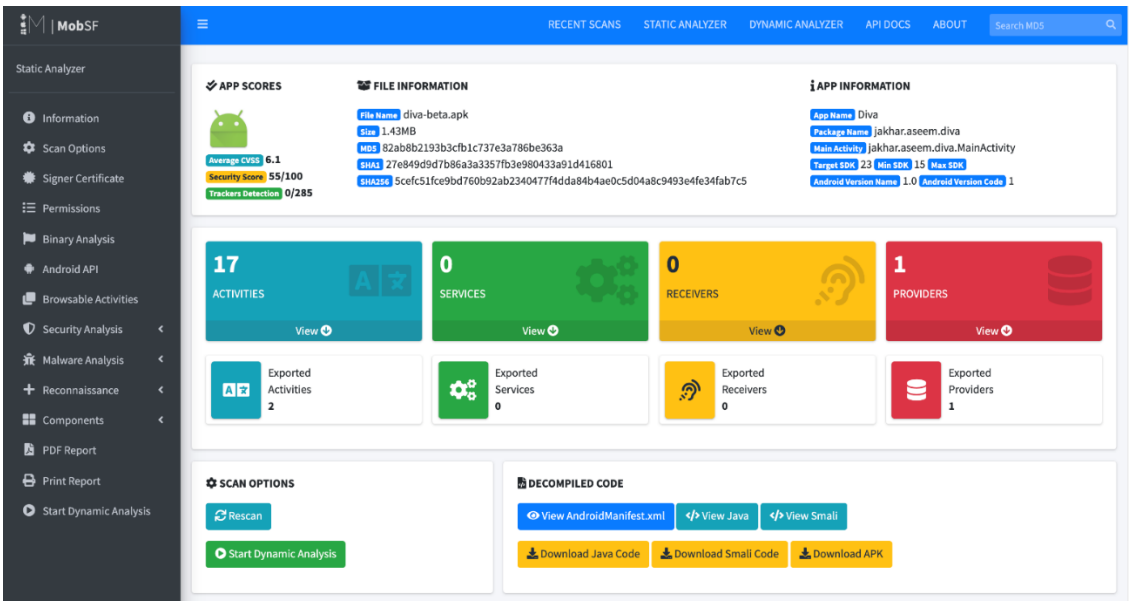


Рисунок 2.2 – Сканування за допомогою MobSF

Це інструмент, який підтримує різні мови програмування, такі як Java, Ruby, Python, JavaScript і PHP. Можна запустити його в одній команді через командний рядок за допомогою CLI CodeClimate.

Code laser також є проектом з відкритим вихідним кодом, який містить набір спеціальних правил для статичного аналізу коду в проектах angular typescript. Статичний аналіз коду можна виконувати на вебсторінках, NativeScript, Ionic та інших платформах.

Crawljax - це відкритий інструмент Java, який використовується для автоматичного сканування і тестування вебдодатків. Він використовує механізм динамічного рейтингу на основі подій Ajax для дослідження фреймворків, що базуються на JavaScript. Цей інструмент автоматично створює діаграму переходів стану DOM на основі подій, що дозволяє автоматизувати багато вебдосліджень і тестувань.

ESLint - це інструмент для визначення шаблонів у коді ECMAScript/Javascript і генерації звітів про них, який має відкритий вихідний код. Він повністю інтегрується з усіма правилами і забезпечує високу гнучкість в налаштуванні. Незважаючи на те, що у нього є рекомендації для початкового використання, правила можна динамічно

завантажувати в будь-який момент. ESLint реалізований на Node.js, що забезпечує легку установку через npm.

Esprima - це стандартний парсер JavaScript з високою продуктивністю, який забезпечує повне синтаксичне дерево. Це дозволяє застосовувати різні статичні аналізи для розуміння коду, такі як візуалізація синтаксису, перевірка коду та редагування з автоматичним завершенням.

Flow - це інструмент з відкритим вихідним кодом, розроблений Facebook, призначений для керування тестами, перевірки статичного стилю та пошуку помилок в коді JavaScript. Flow використовує статичну типізацію JavaScript для покращення ефективності та узгодженості нового коду.

JavaScript Lint - це інструмент для перевірки коду JavaScript з відкритим вихідним кодом, який зосереджений на синтаксичних правилах JSLint. Він надає сувору структуру для перевірки синтаксису JavaScript і надає поради щодо підозрілих дій.

Jscent - це сканер, який розпізнає "запахи" коду, тобто потенційні проблеми, які можуть виникнути в програмі. Його звіти містять детальний опис всіх "запахів" коду, що полегшує їх аналіз і виправлення.

JSDeodorant - це інструмент для оцінки коду JavaScript з відкритим вихідним кодом, який визначає структури класів симуляції JavaScript для популярних романів, блогів і тенденцій. Він базується на дереві AST Closure Compiler і використовує аналіз потоку даних.

JSHint є відкритим фреймворком для перевірки коду JavaScript та впровадження кодувальних стандартів для команди розробників. Проект Дугласа Крокфорда можна аналізувати онлайн на вебсайті JSHint.

JSLint є платформою з відкритим вихідним кодом, яка виявляє помилки у JavaScript. Це безкоштовний інструмент для покращення продуктивності вихідного коду. Онлайн-аналіз коду JavaScript доступний на вебсайті JSLint.

JSPrime - це платформа для перевірки статичних уразливостей у коді JS. Інструмент, заснований на популярному парсері ECMAScript від EsprimA, є дуже компактним та легким у використанні.

Nocuous - це інструмент для перегляду статичного коду JavaScript і TypeScript.

PHP CodeSniffer - це серія скриптів для PHP, які допомагають виявляти порушення кодувальних стандартів у PHP, JavaScript і CSS. PHP CodeSniffer є важливим інструментом програмування, який забезпечує безпеку та сумісність коду.

2.1.2 Інструменти динамічного аналізу вебсторінки

Найпоширенішими інструментами для динамічного аналізу є [15]:

Автоматизоване тестування та тестування продуктивності Smartbear, відоме як "Розумний ведмідь", пропонує широкий набір інструментів, таких як Test Complete, BitBar, Load Ninja та Cucumber, які допоможуть забезпечити високу якість програмного забезпечення. Ці інструменти гарантують, що програма функціональна та безпечна (рис. 2.3) [15].

Дане програмне забезпечення дозволяє проводити сканування на наявність таких атак як XSS, HTTP Method Fuzzing, SQL Injection, XPath Injection, Invalid Types тощо. Після сканування програма надає звіт, де вказані всі недоліки та пропозиції, які дозволять покращити безпеку вебдодатку.

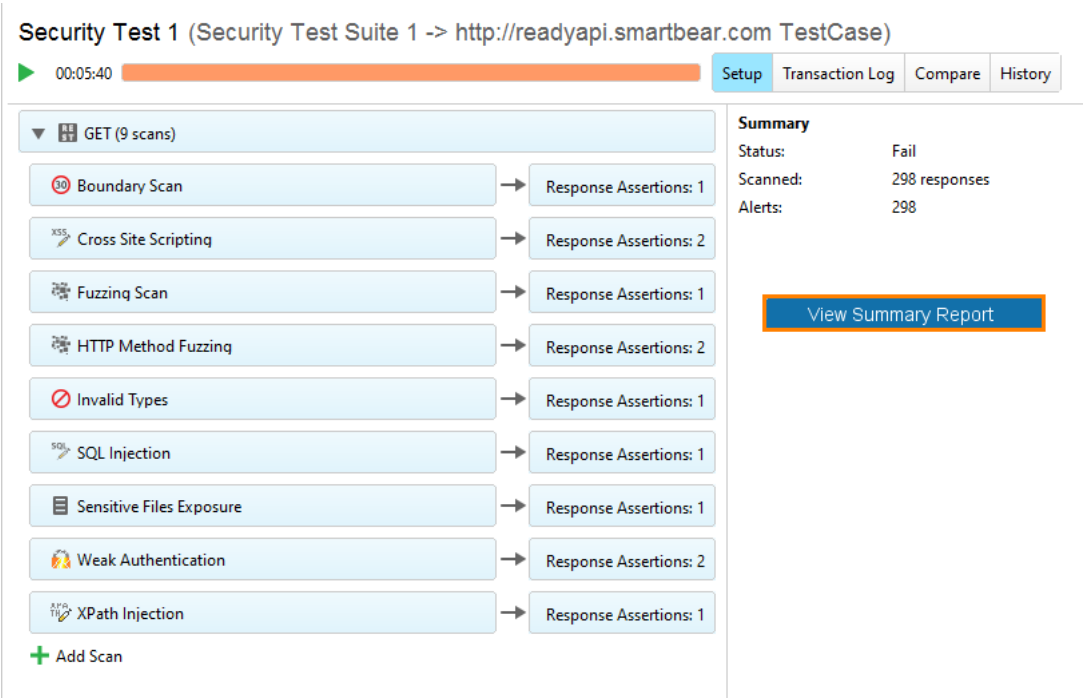


Рисунок 2.3 – Сканування за допомогою SmartBear

Інструмент gcov, який можна використовувати з GCC, допомагає перевірити покриття коду вашої програми. gcov створює файл журналу, який показує, скільки разів кожен рядок вихідного файлу було виконано, що допомагає визначити області коду, які потребують оптимізації [15].

Code Pulse - це безкоштовний інструмент для перевірки коду в реальному часі, який автоматично виявляє інформацію про покриття під час проведення тестів. Code Pulse надає візуальну інформацію про охоплення, щоб легко зрозуміти, які частини програми були охоплені [15].

Програма показує, як код вебдодатку, що сканується в режимі реального часу протистоїть тестуванню на проникнення. Сканер поділяє код на частини і вказує, які з них охоплені, а які ні (рис. 2.4).

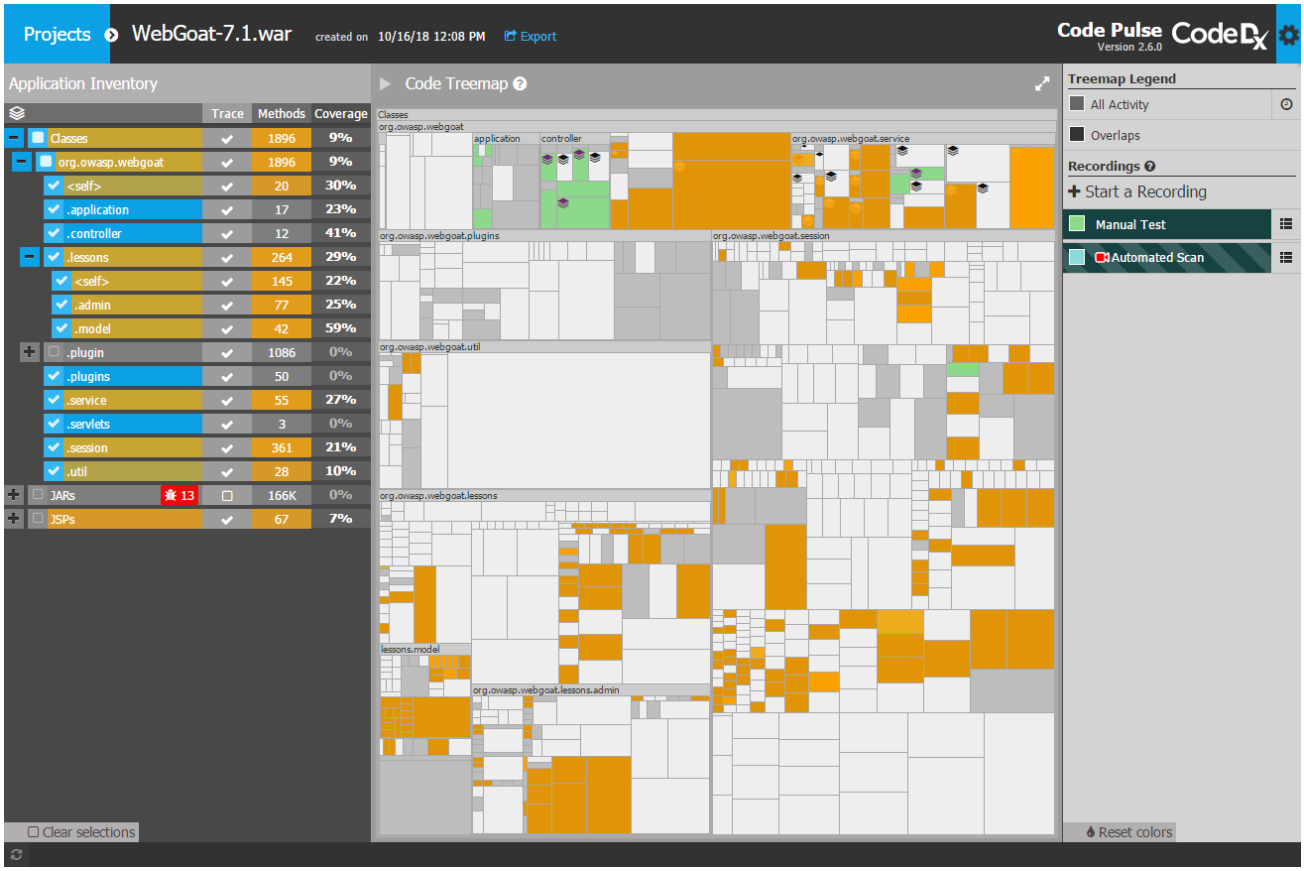


Рисунок 2.4 – Сканування за допомогою Code Pulse

Enlightn - це сканер уразливостей, призначений для програм Laravel PHP, який поєднує різні методи аналізу для виявлення вразливостей. Enlightn сканує ваш код

програми Laravel та надає рекомендації щодо покращення продуктивності та безпеки, допомагаючи виявити проблеми на ранніх етапах [15].

Enlightn представлений як додаток у командній строці. Програма проводить перевірки на наявність різних вразливостей та допоміжних сервісів, які може використовувати вебдодаток, що сканується (рис. 2.5).

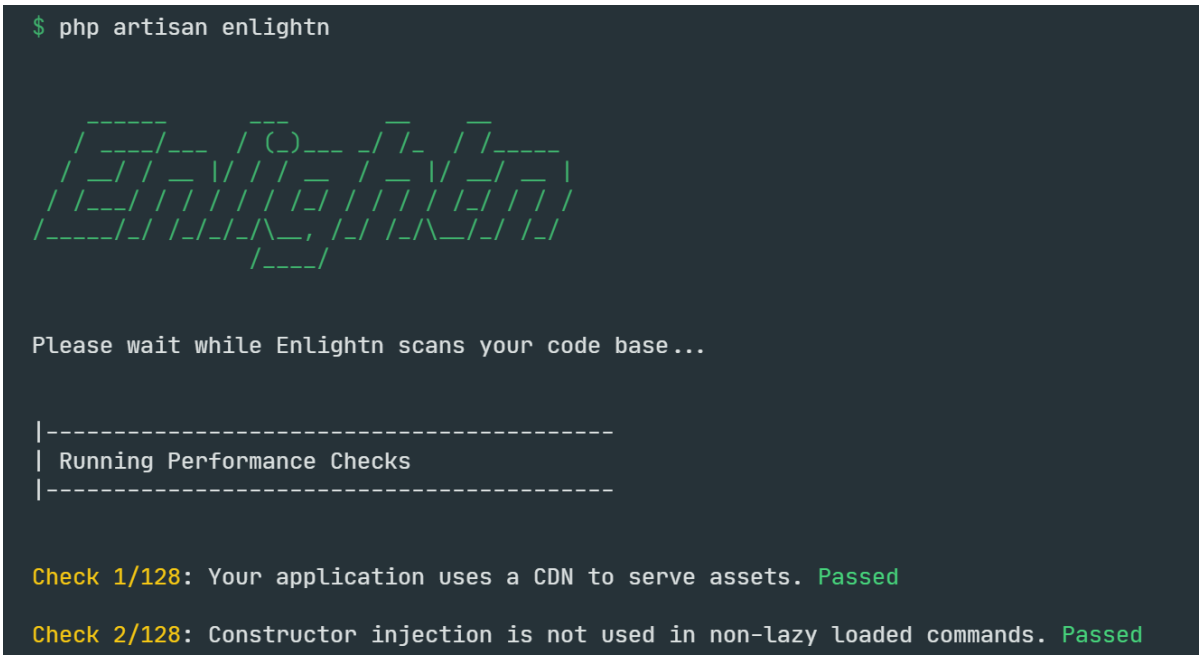


Рисунок 2.5 – Сканування за допомогою Enlightn

IROH.JS відомий як Iroh, надає можливість відстежувати, перехоплювати та маніпулювати даними у вашому коді під час його виконання. Ви можете змінювати типи, параметри, значення та інші аспекти коду, що дозволяє вам здійснювати більш глибокий аналіз та оптимізацію вашого програмного забезпечення [15].

Valgrind як основа для створення інших інструментів динамічного аналізу, дозволяє виявляти помилки керування пам’яттю та потоками, а також детально профілювати програми. Valgrind надає можливість створення нових інструментів, що допомагають забезпечити високу якість вашого програмного коду [15].

Сканер допомагає виявити помилки у коді, що тестується. Головною функцією даного сканера - це робота з пам’яттю, а саме, як поділяється пам’ять, на скільки вона використовується тощо (рис. 2.6).

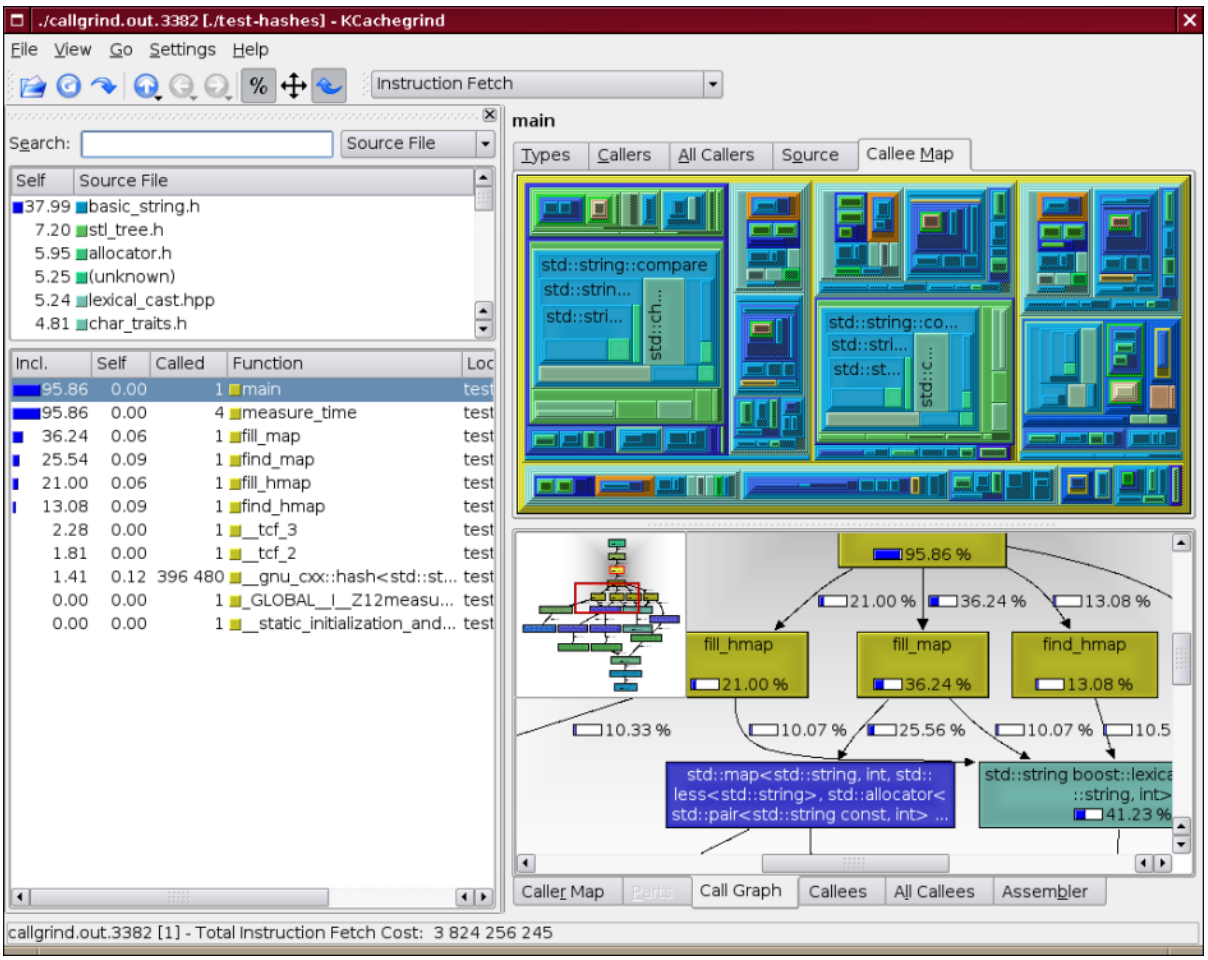


Рисунок 2.6 – Сканування за допомогою Valgrind

CHAP відомий як Chap, аналізує основні файли вашого коду на наявність витоків пам’яті та інші проблеми. Цей інтерактивний інструмент допомагає виявити та виправити проблеми безпеки ще до того, як вони стануть критичними [15].

Wasabi спрямований на динамічний аналіз WebAssembly, забезпечує простий у користуванні API для реалізації складних аналізів. Wasabi базується на бінарному інструментуванні, що дозволяє контролювати поведінку низькорівневого коду у вашому програмному забезпеченні [15].

HCL AppScan Standard є інструментом динамічного тестування безпеки додатків, призначеним для експертів з безпеки та тестувальників. AppScan автоматично сканує програму на наявність вразливостей та пропонує рішення для їх виправлення (рис. 2.7) [15].

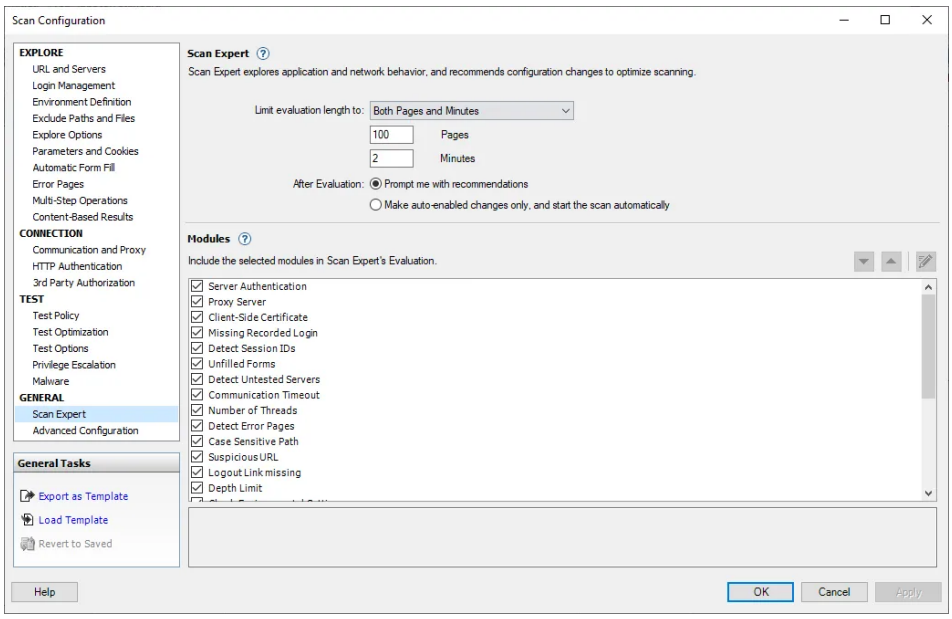


Рисунок 2.7 – Сканування за допомогою HCL AppScan Standard

2.1.3 Інструменти, які використовуються для цільових типів атак

Окрім даних інструментів для статичного та динамічного аналізу є спеціальні інструменти для аналізу вразливостей до поширених атак на вебдодатки.

Acunetix Vulnerability Scanner - це комплексний і автоматизований інструмент для виявлення вразливостей в програмному забезпеченні. Він використовує методи аналізу Black-Box і Gray-Box для виявлення різноманітних проблем безпеки та може бути розгорнутий як в хмарі, так і на стороні клієнта. Acunetix здатний ідентифікувати і повідомляти про різноманітні вразливості в програмах, що побудовані на різних платформах, таких як WordPress, PHP, ASP.NET, Java Framework, Ruby on Rails та інші [18].

Інструмент має широкий набір можливостей як для автоматизованого, так і для ручного тестування, що дозволяє оцінити та усунути виявлені проблеми безпеки. Система Acunetix розрахована на роботу з багатьма користувачами і забезпечує доступ лише до необхідних ресурсів, що сприяє командній гнучкості та підвищує продуктивність (рис. 2.8) [18].

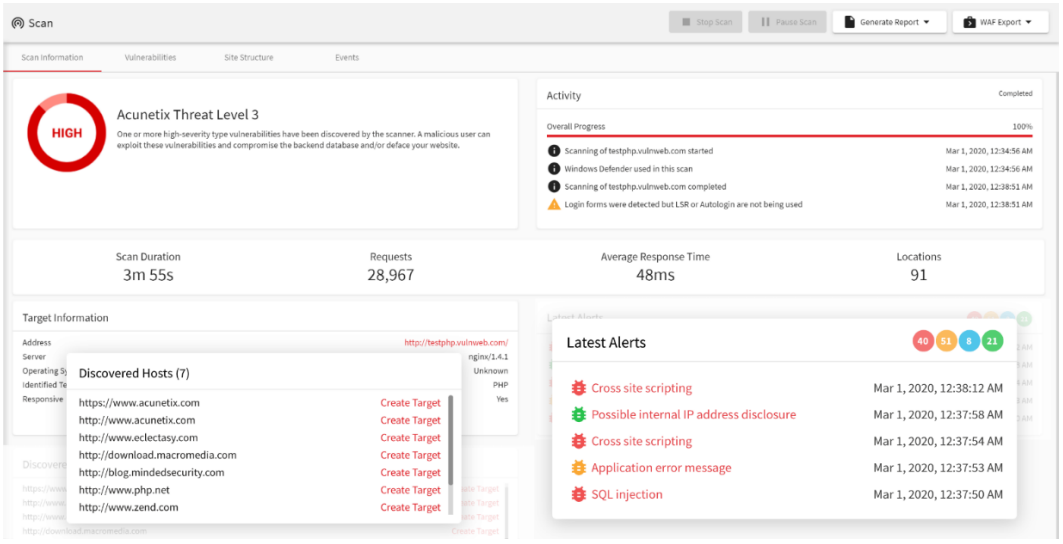


Рисунок 2.8 – Сканування за допомогою Acunetix Vulnerability Scanner

AppSpider - це рішення для динамічного тестування безпеки додатків, яке здатне сканувати веб та мобільні додатки на предмет наявності вразливостей [19].

Основною технологією, що використовується в AppSpider, є універсальний перекладач, який адаптується до новітніх технологій, таких як AJAX, HTML5 і JSON, що застосовуються у сучасних веб та мобільних додатках, і проводить сканування як традиційних, так і новітніх програм [19].

AppSpider доступний для використання на місці, в хмарі або як керований сервіс, і дозволяє ефективно управляти програмою безпеки додатків, забезпечуючи ретельний аналіз, повне охоплення додатків і складні методології атак (рис. 2.9) [19].

Основні переваги AppSpider включають [19]:

- широке охоплення;
- розширена аутентифікація;
- інтеграції;
- інтерактивні звіти;
- розподілений і масштабований функціонал;
- централізований контроль;
- постійний моніторинг вебсайтів;
- наскрізне тестування API, що базуються на специфікації OpenAPI (раніше відомій як Swagger).

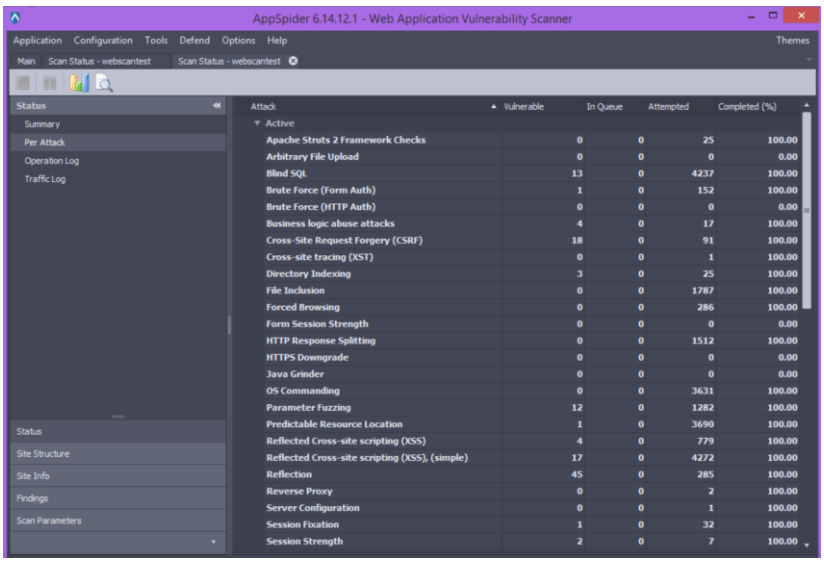


Рисунок 2.9 – Сканування за допомогою AppSpider

OWASP ZAP - це інструмент, який дуже простий у використанні для проведення тестів на проникнення програмою, а також для виявлення вразливостей у вебдодатках. Ця програма призначена як для досвідчених користувачів у сфері ІБ, таких як розробники та QA [21].

OWASP ZAP проводить тестування на проникнення вебдодатку та дозволяє виявляти такі атаки як SQL injection, XSS, clickjacking тощо (рис. 2.10).

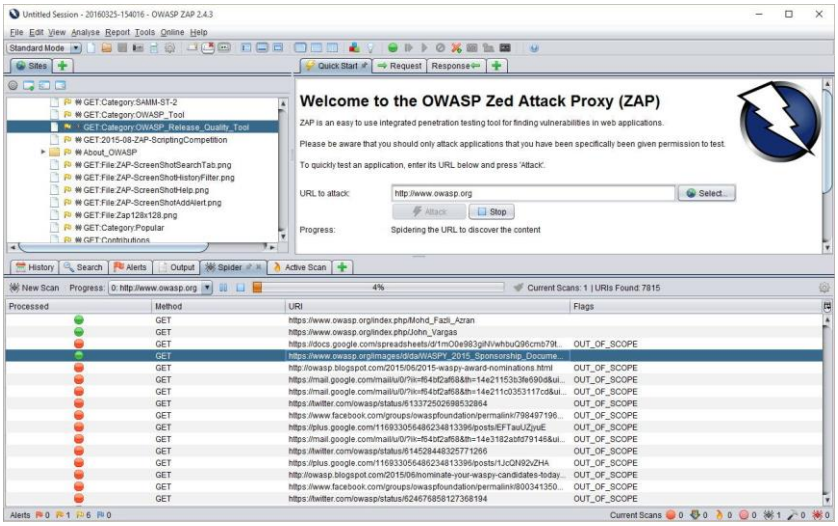


Рисунок 2.10 – Сканування за допомогою OWASP ZAP

Nmap чи Network Mapper - це відкритий інструмент, призначений для вивчення мережі та перевірки її безпеки. Початково він був розроблений для швидкого

сканування великих мереж, але також ефективно застосовується для аналізу окремих цілей. Nmap використовує IP пакети у специфічний спосіб для визначення доступних хостів у мережі, надаючи інформацію про надані ними послуги (назву та версію програм), операційні системи та їх версії, типи використаних пакетних фільтрів/брандмауерів та інші характеристики. Nmap часто використовується для оцінки безпеки, а також для контролю структури мережі та управління розкладами запуску служб та обліку часу роботи хостів або служб [22].

Nmap дозволяє сканувати порти системи, їх характеристики та встановлені сервіси (ssh, ftp, http тощо) (рис. 2.11).

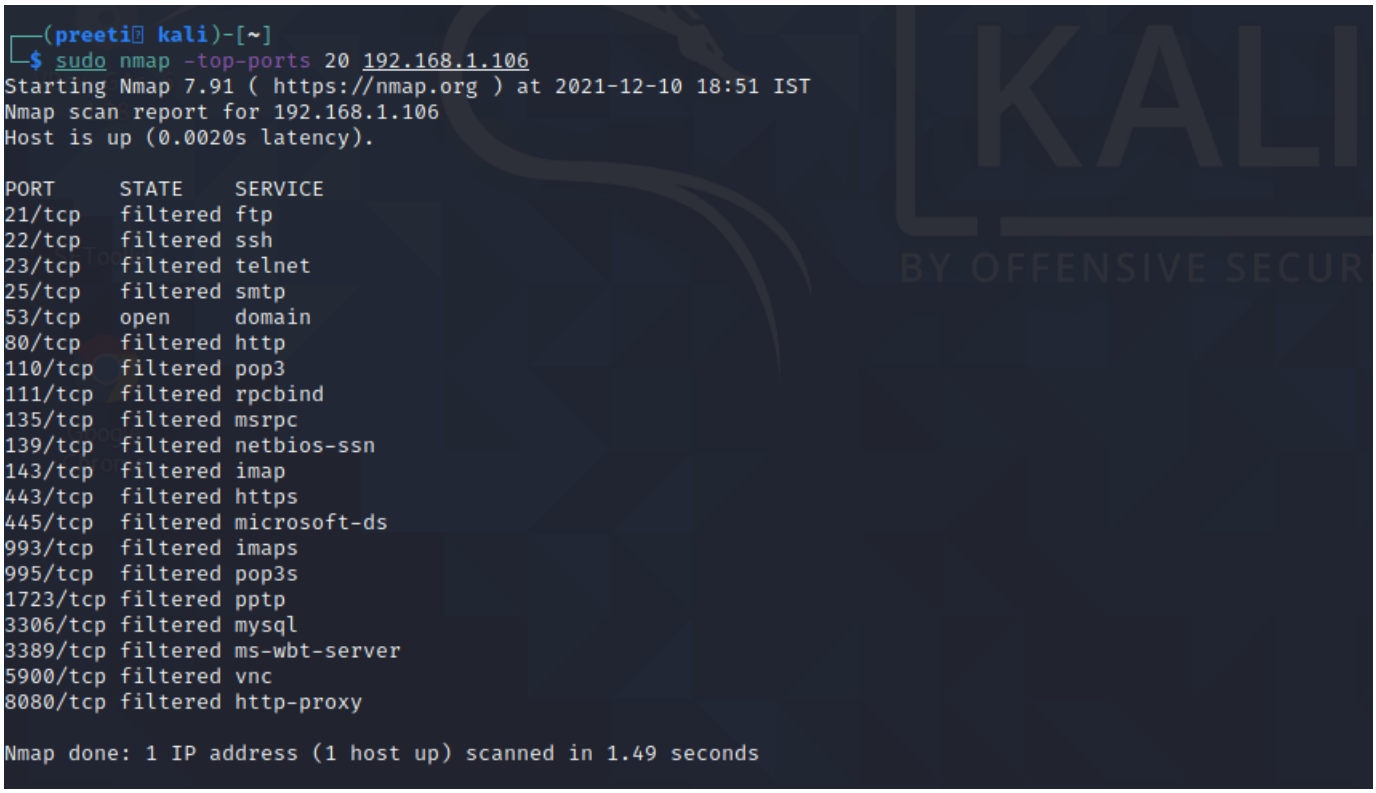


Рисунок 2.11 – Сканування за допомогою Nmap

Metasploit Framework — це дуже потужний інструмент, який використовують як кіберзлочинці, так і етичні хакери для виявлення системних уразливостей у мережах та на серверах (рис. 2.12). Оскільки це фреймворк з відкритим кодом, його можна легко налаштувати та використовувати на більшості операційних систем [23].

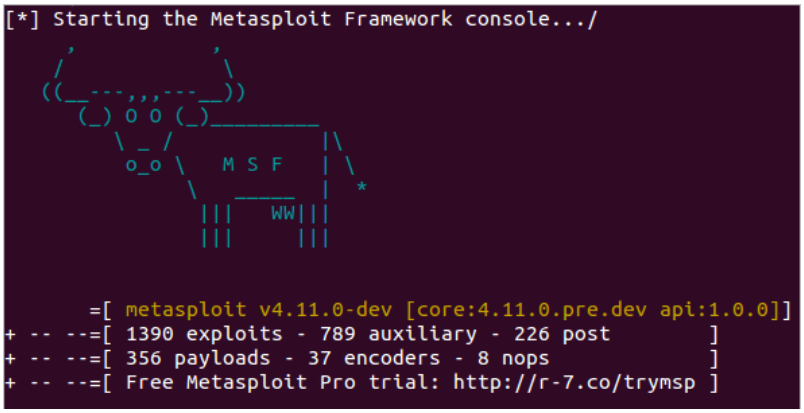


Рисунок 2.12 – Використання Metasploit

Команда тестувальників може використовувати Metasploit для застосування готового або створення спеціального коду та його впровадження у мережу для пошуку слабких місць. Це ще один метод полювання на загрози, де після виявлення та документування недоліків інформація може бути використана для усунення системних вразливостей та визначення пріоритетів у вирішенні проблем [23].

Wireshark - це спеціальне програмне забезпечення з відкритим вихідним кодом, яке дозволяє аналізувати трафік в мережі (рис. 2.13). Wireshark дозволяє перевіряти, захоплювати та переглядати пакети даних, які передаються у комп’ютерних мережах. Wireshark надає можливість вивчення протоколів, виявлення недоліків в мережі. Використовуючи різні підходи та протоколи можна досконало проаналізувати мережу, виявити слабкі місця та запобігти шкідливим діям зловмисників [26].

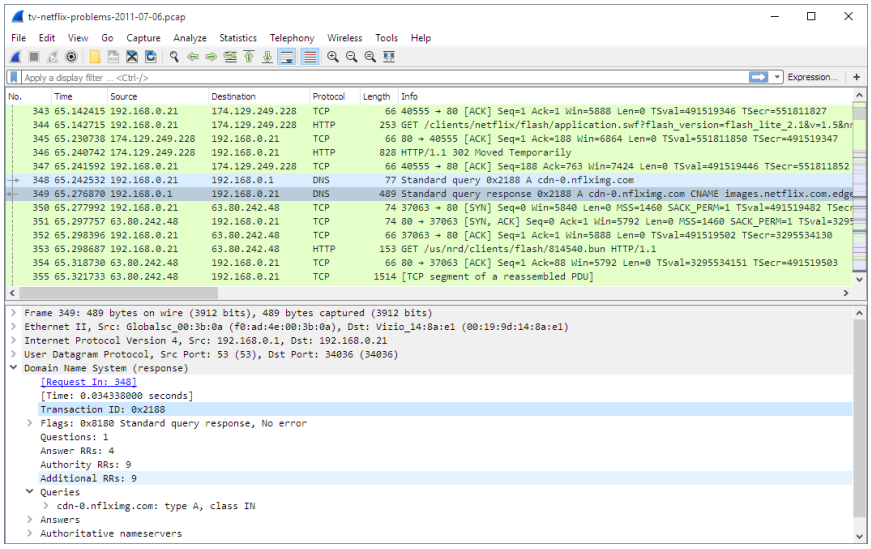


Рисунок 2.13 – Сканування мереж за допомогою Wireshark

SQLmap - це інструмент для тестування на проникнення, який автоматизує виявлення та експлуатацію SQL-ін’єкцій та захоплення серверів баз даних. Інструмент має вбудований механізм виявлення з спеціалізованими можливостями для досвідчених тестерів на проникнення та широким спектром параметрів для отримання детального звіту з тестування на проникнення. SQLmap може використовуватися зловмисниками для проведення SQL-ін’єкцій на потрібні додатки за допомогою різних методів, таких як логічні значення, часові проміжки, помилки, UNION-запити, стековані запити та позасмугове впровадження (рис. 2.14) [24, 25].

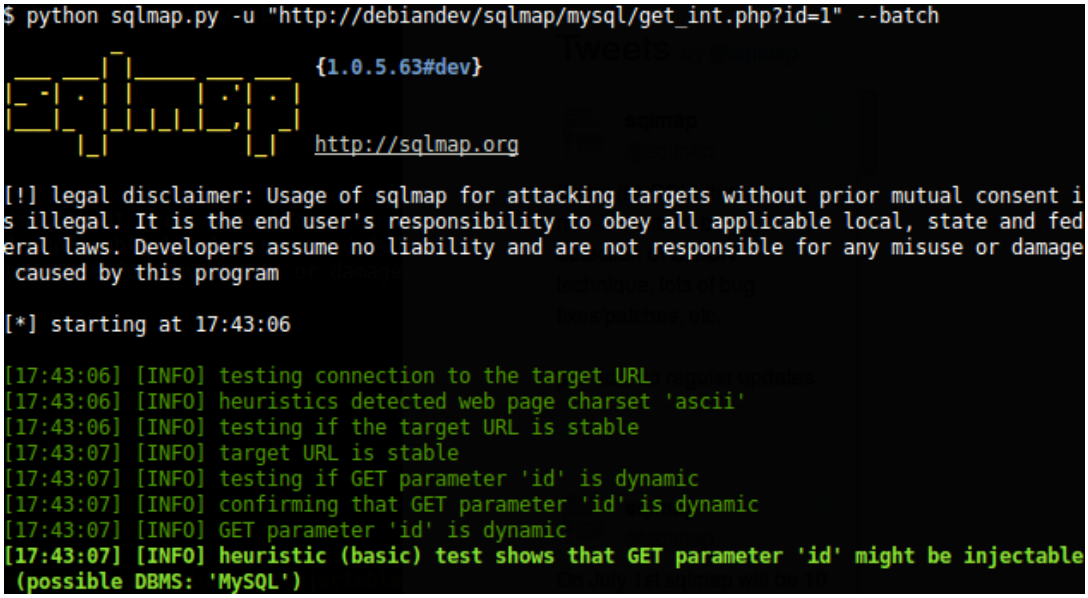


Рисунок 2.14 – Сканування за допомогою SQLmap

Ettercap — це інструмент з відкритим вихідним кодом, який призначений для підтримки атак типу "людина посередині" в мережах. Він може перехоплювати пакети та записувати їх у мережу, а також перенаправляти та змінювати дані майже в реальному часі. Ettercap також може використовуватися для аналізу необхідного протоколу для детального дослідження мережевого трафіку [27].

2.2 Методології аналізу вебдодатків на вразливості

Дослідження вебдодатків на вразливості сьогодні - це важливе завдання, яке дозволить вебпрограмам функціонувати належним чином. Дане тестування

виконують розробники та тестувальники ПЗ, які зазвичай використовують вже наявні та перевірені часом методики. До таких методологій можна віднести [28]:

- Open Source Security Testing Methodology Manual (OSSTMM);
- Open Web Application Security Project (OWASP);
- Web Application Security Consortium Threat Classification (WASC-TC);
- Penetration Testing Execution Standard (PTES);
- Information Systems Security Assessment Framework (ISSAF);
- PCI Penetration Testing Guide;
- NIST Special Publication 800-115;
- MITRE ATT&CK.

2.2.1 Open Source Security Testing Methodology Manual (OSSTMM)

The Open Source Security Testing Methodology Manual, також відомий як OSSTMM, є методологією для проведення тестів на безпеку. Розроблена Пітом Герцогом у 2000 році, яка пройшла рецензування та підтримується Інститутом безпеки та відкритих методологій (ISECOM) [29].

Даний документ оновлюється кожні приблизно шість місяців, щоб бути адаптованим до сучасного стану тестування безпеки. Головною метою ISECOM у контексті OSSTMM - це надання наукового підходу для коректного розуміння безпекових операцій. Він може використовуватись для проведення тестів на проникнення, етичного взлому та інших випробувань безпеки. ISECOM використовує достовірні факти, щоб забезпечити, що організації, які використовують OSSTMM для тестування на проникнення, можуть приймати обґрунтовані рішення [30].

Ця спільнота активно ділиться практичними знаннями з безпеки, здійснює дослідження та надає сертифікацію. OSSTMM розглядає різні аспекти операційної безпеки у п’яти основних сферах: інформаційна безпека, безпека процесів, безпека інтернет-технологій, безпека зв’язку та фізична безпека [30].

В Open Source Security Testing Methodology Manual входять такі розділи [30]:

- Operational Security Metrics;

- Trust Analysis;
- Work Flow;
- Human Security Testing;
- Physical Security Testing;
- Wireless Security Testing;
- Telecommunications Security Testing;
- Data Networks Security Testing;
- Compliance Regulations;
- Reporting with the STAR (Security Test Audit Report).

OSSTMM працює за структурованим підходом, приділяючи особливу увагу операційній безпеці та взаємодії. Він проходить через спеціальний цикл тестування [30]:

1. Підготовка: цей етап включає визначення обсягу робіт, з'ясування контексту системи, ідентифікацію ризику та отримання необхідних дозволів.

2. Оцінка: на цьому етапі проводиться аудит систем контролю та процедур, сканування систем та оцінка їх поточного рівня безпеки.

3. Тестування: на цьому етапі активно досліджуються вразливості, експлойти та слабкі місця в системі.

4. Звітність: цей етап передбачає документування результатів тестування, визначення вразливостей та надання рекомендацій щодо подальших заходів для підвищення безпеки.

5. Оптимізація: на останньому етапі проводиться оптимізація процесу тестування, включаючи перевірку виправлень, повторне тестування при необхідності та постійне вдосконалення.

OSSTMM виступає як орієнтир для професіоналів у галузі безпеки, етичних хакерів, аудиторів та організацій з метою створення безпечного середовища. Широке використання цієї методології відбувається у проведенні тестів на проникнення, IT-аудиті та оцінці ризиків для підтвердження ефективності заходів безпеки [29].

Наприклад, компанії можуть використовувати OSSTMM для проведення тестування на проникнення у своїй інфраструктурі. Підприємство керуватиметься

цією методологією, розпочинаючи з підготовки та завершуючи оптимізацією, що документує кожен етап для забезпечення прозорості та відповідальності. Результати тестування будуть об’єктивно вимірюватися за допомогою показників RAV і SAFE, які надають наукову підставу для оцінки рівня безпеки та визначення подальших заходів з покращення [30].

Розглянемо приклади використання OSSTMM. У реальному світі, декілька організацій у різних галузях успішно скористалися OSSTMM для проведення тестування безпеки, що підтверджує широкий спектр його застосування та ефективність [30].

Наприклад, фінансова компанія залучила OSSTMM для проведення комплексного аудиту безпеки. Маючи обширну мережу взаємопов’язаних систем і потребу в суворій конфіденційності, організація вирішила використати OSSTMM для аудиту безпеки. Цей процес допоміг виявити кілька вразливостей в їхній IT і фізичній безпеці, які раніше залишалися непоміченими. Після вирішення цих проблем, компанія відзначила помітне зменшення випадків порушень безпеки та інцидентів [30].

У іншому випадку, постачальник медичних послуг використовував OSSTMM для забезпечення відповідності правилам з охорони здоров’я та захисту конфіденційності даних пацієнтів. Тестування, проведене за допомогою OSSTMM, виявило недоліки в безпеці даних та управлінні процесами. В результаті, постачальник медичних послуг зміг покращити свої процедури з безпеки даних, що сприяло підвищенню довіри як до пацієнтів, так і до партнерів [30].

Ці приклади ілюструють потенціал OSSTMM у виявленні вразливостей та підвищенні рівня безпеки в різних галузях.

Розглянемо OSSTMM та відповідність нормативними вимогами. Це важливий аспект організаційної безпеки. OSSTMM може сприяти цьому, надаючи надійну структуру для виявлення вразливостей та покращення засобів контролю безпеки [30].

Наприклад, ISO 27001, міжнародний стандарт для систем управління інформаційною безпекою, вимагає регулярних перевірок безпеки. Структурований і

комплексний характер OSSTMM робить його ефективною методологією для цих аудитів, забезпечуючи ретельну оцінку всіх аспектів безпеки [30].

Також, акцент OSSTMM на захисті даних узгоджується з вимогами GDPR щодо безпеки даних. Виявлення та усунення вразливостей допомагає організаціям демонструвати свою прихильність до захисту даних та відповідність GDPR [30].

У контексті HIPAA, що стосується захисту медичної інформації, OSSTMM може допомогти виявити потенційні загрози для даних пацієнтів і надати корисну інформацію для посилення заходів безпеки, що сприяє досягненню відповідності HIPAA [30].

Розглянемо відгуки та обмеження OSSTMM. Незважаючи на свою ефективність, OSSTMM має деякі потенційні обмеження. Першим недоліком є його складність. Посібник має обширний і технічно складний зміст, що може ускладнювати його застосування для організацій з обмеженим досвідом у сфері безпеки [30].

Крім того, хоча OSSTMM надає детальну методологію для тестування, він не включає конкретних заходів щодо усунення виявлених вразливостей. Відповідальність за розробку та впровадження таких заходів покладається на організацію, що може бути викликом для деяких компаній [30].

Хоча OSSTMM акцентується на ручному тестуванні, цей підхід може займати більше часу порівняно з автоматизованими інструментами, особливо для великомасштабних інфраструктур [30].

Розглянемо порівняння OSSTMM з іншими методологіями безпеки. Порівняння OSSTMM з іншими методологіями, такими як стандарт виконання тестування на проникнення (PTES) та рекомендації щодо тестування OWASP, дає корисні відомості.

PTES, подібно до OSSTMM, надає структурований підхід до тестування на проникнення. Однак, PTES більш зосереджений на технічних аспектах тестування, тоді як OSSTMM охоплює ширший спектр аспектів безпеки, включаючи операційні, людські та фізичні аспекти [30].

Методологія тестування OWASP спеціалізується на безпеці вебдодатків, надаючи комплексний підхід для виявлення вразливостей. У порівнянні, OSSTMM охоплює ширший спектр аспектів безпеки, не обмежуючись лише вебдодатками [30].

Таким чином, вибір між цими методологіями залежить від конкретних потреб безпеки та контексту організації.

Розглянемо майбутнє OSSTMM. У зв'язку із швидкими змінами у кібербезпеці, OSSTMM ймовірно буде продовжувати розвиватися та адаптуватися до нових викликів. Можливість більшої інтеграції з автоматизованими інструментами може прискорити процес тестування, зберігаючи при цьому ретельність ручного тестування [30].

Оскільки правила щодо конфіденційності даних стають суворішими, OSSTMM може включати більш докладні вказівки щодо перевірки конфіденційності. Також можливе зосередження на нових сферах безпеки, таких як безпека Інтернету речей та ШІ [30].

Завдяки своїй простоті, OSSTMM може продовжувати вдосконалюватися та адаптуватися до нових загроз безпеки та змін у технологіях, залишаючись важливим інструментом у сфері кібербезпеки у майбутньому [30].

2.2.2 Open Web Application Security Project (OWASP)

Open Web Application Security Project (OWASP) - некомерційна організація, яка працює над підвищенням рівня безпеки програмного забезпечення. Місія фонду полягає у посиленні кібербезпеки програмного забезпечення. У цьому йому сприяє активна міжнародна спільнота розробників, фахівців та ентузіастів. Мережа фонду налічує понад 250 представництв та десятки тисяч учасників по всьому світу. Щорічно проводяться масштабні освітні та навчальні конференції. Спільнота надає організаціям платформу для створення, розробки, впровадження, експлуатації та підтримки надійних програмних продуктів. Важливою особливістю є те, що всі проєкти, інструменти, документація та конференції доступні безкоштовно та відкриті для всіх, хто прагне до підвищення рівня безпеки програмного забезпечення. [31, 32].

Спільнота OWASP прагне ніколи не рекламувати або підтримувати певні продукти або послуги у сфері кібербезпеки, хоча це може здатися нелогічним. Замість цього, компаніям рекомендується інвестувати у засоби безпеки, які є ефективними та надійними, інакше вони ризикують залишитися уразливими перед сучасними загрозами [31].

Основна мета OWASP полягає в тому, щоб звернути увагу на найбільші загрози у сфері безпеки, з якими ми стикаємося сьогодні. Якби вони приймали оплату за рекламу чи підтримку, це підірвало б їхню об'єктивність та надійність, оскільки б користувачі не могли б бути впевнені, чи рекомендується продукт насправді через його якість, чи через оплату [33].

Захищене кодування OWASP визначає вразливі місця безпеки як дефекти або слабкі моменти в програмному коді, що виникають в результаті недоліків у конструкції чи помилок реалізації. Ці недоліки можуть бути використані зловмисниками для завдання шкоди користувачам програмного забезпечення, власникам або іншим об'єктам, що використовують додаток. OWASP наводить різні категорії вразливостей на своєму вебсайті, включаючи аутентифікацію, доступність, якість коду, обробку помилок, загальні логічні помилки, перевірку введених даних та помилки протоколу [33].

Деякі категорії уразливостей OWASP безпосередньо пов'язані з якістю та розробкою програми. Ці вразливості можна виявити за допомогою статичного та структурного аналізу якості вихідного коду. З ростом розміру та складності системи, а також з участю кількох джерел розробки, які працюють над одним кодом, стає критично важливим перевіряти відповідність стандартам безпеки та якості протягом усього життєвого циклу проекту [33].

Переваги OWASP [34]:

- відкритий вихідний код: безкоштовний та доступний, що робить його доступним для будь-кого;
- підтримка різних мов програмування: підтримує широкий спектр мов програмування, включаючи Java, .NET і Python, що робить його корисним для різноманітних програм;

- регулярні оновлення: регулярно оновлюється новою інформацією про вразливості, що допомагає визначити найновіші загрози.

Недоліки OWASP [34]:

- обмежена база відомих вразливостей: може виявити лише вразливості, які вже відомі та перераховані в NVD (National Vulnerability Database). Якщо в одній із ваших залежностей буде виявлено нову вразливість, яку ще не додано до NVD, OWASP не зможе її ідентифікувати;
- хибні спрацьовування: може генерувати хибні спрацьовування, оголошуючи про наявність вразливостей, яких насправді немає. Це може бути спричинено неправильною чи застарілою інформацією в NVD або розбіжностями у способі, яким та програма інтерпретують ідентифікатор CPE залежності;
- потреба у ручному втручанні: може виявляти вразливості, але не виправляє їх автоматично. Користувач сам вирішує, як усунути вразливості, що може вимагати ручного втручання.

2.2.3 Web Application Security Consortium Threat Classification (WASC-TC)

Організація WASC (The Web Application Security Consortium) раніше відома своєю активною роботою у сфері стандартизації безпеки вебдодатків. Проте, наразі організація не проявляє активності. Частина учасників WASC також взяла участь у розробці проектів OWASP [35].

Класифікація загроз WASC є результатом спільної роботи з уточнення та організації загроз безпеці вебсайтів. Члени Консорціуму безпеки вебдодатків створили цей проект з метою розробки та просування галузевих стандартів термінології для опису цих проблем. Розробники програм, спеціалісти з безпеки, постачальники програмного забезпечення та аудитори відповідності отримають можливість користуватися узгодженою мовою для розгляду питань, пов’язаних з безпекою вебдодатків [35, 36].

WASC Threat Classification (далі - TC) визначає різновиди атак та слабкості, які можуть призвести до порушення безпеки вебдодатків, їхніх даних чи користувачів.

Перша версія класифікації була представлена у 2004 році, друга - у 2010 році, і залишається останньою наразі. У TC атаки та слабкості поділені на три фази розробки (проектування, реалізація, впровадження). Всього в переліку описано 34 типи атак та 15 типів слабкостей. Для кожного елементу подається опис, приклади та корисні посилання [35].

Класифікацію розробили експерти з безпеки вебдодатків. Під час розробки класифікації, всі правки обговорювалися через електронну пошту. Кожен розділ проекту обговорювався протягом тижнів, поки всі учасники не дійшли до одного висновку. Однак не зовсім зрозуміло, як з методичної точки зору група експертів забезпечувала повноту та несуперечливість класифікації, крім як у формі експертної згоди на розроблені документи [35].

WASC TC здобув широке визнання як цінний довідник, який активно використовувався у наукових роботах, книгах та звітах. Цей проект відігравав значну роль у процесі оцінки безпеки вебдодатків, слугуючи контролем або планом тестування. Окрім того, WASC TC використовувався для систематизації даних про недоліки та вразливості, виявлені у вебзастосунках. Однак, важливо зазначити, що сфера безпеки вебдодатків динамічно розвивається, а оновлення WASC Threat Classification не проводились з 2010 року. [35].

Класифікація загроз версії 2.0 розглядає різновиди атак та вразливості, які можуть призвести до порушення безпеки вебсайтів, їх даних або користувачів. Головною метою цього документу є надання довідкової інформації для кожного конкретного типу атаки або вразливості, включаючи приклади та корисні матеріали. Цей документ широко використовується різними організаціями і, зазвичай, застосовується на різних етапах роботи з безпекою [35].

2.2.4 Penetration Testing Execution Standard (PTES)

Penetration Testing Execution Standard (PTES) є методом тестування на проникнення, розробленим практиками інформаційної безпеки, з метою задоволення потреб у повній і актуальній стандартизації у сфері тестування на проникнення. Він

також намагається проінформувати підприємства про те, що їм слід очікувати від тестування на проникнення, і допомогти їм у визначенні обсягу робіт і веденні переговорів щодо успішних проектів [37].

Тестування на проникнення - це процес, під час якого організації перевіряють свій рівень безпеки, симулюючи атаки в реальному світі. Основний принцип полягає в тому, щоб визначити та усунути вразливості перед тим, як їх можуть використати зловмисники. Існують різні методи проведення тестування на проникнення, і вибір залежить від конкретних потреб та цілей організації [37].

Однак немає універсального підходу до тестування на проникнення. Penetration Testing Execution Standard (PTES) є відмінним інструментом, який визначає стандартизовану методологію для проведення тестування на проникнення. Він включає значущі практики для кожної фази процесу, починаючи від планування та завершуючи створенням звіту. Далі буде розглянуто основні аспекти PTES та як його можна використовувати для ефективного проведення тестування програм [37].

PTES описує тестування на проникнення у семи основних розділах [37, 38]:

- Pre-engagement Interactions (Попередні взаємодії): це підготовчий етап для тестування на проникнення. Тут йдеться про затвердження документів та інструментів, необхідних для тестування;
- Intelligence gathering (Збір інформації): у цій фазі отримуються відомості про цільову систему зовнішніми джерелами, такими як соціальні медіа, офіційні записи тощо. Цей етап використовується за методом відкритого джерела інформації (OSINT);
- Threat Modelling (Моделювання загроз): це процес оптимізації мережі безпеки шляхом визначення цілей та вразливостей, і встановлення контрзаходів для запобігання чи пом’якшення наслідків загроз системі. У звичайному тестуванні на проникнення цей етап пропускається;
- Vulnerability Analysis: на цьому етапі виявляються та перевіряються вразливості;
- Exploitation (Експлуатація): на цьому етапі дослідник намагається скористатися виявленими та перевіреними вразливостями для першої атаки системи;

- Post Exploitation (Післяексплуатаційна діяльність): це етап, коли отримується контроль над цільовою системою та збирається інформація;
- Reporting (Звітність): документується весь процес у формі звіту для клієнта.

Penetration Testing Execution Standard (PTES) - це перелік етапів, які слід враховувати під час проведення тестування на проникнення. Він включає в себе високорівневе керівництво з питань, щодо видів тестів, які слід виконати, а також конкретні деталі кожного тесту. PTES надає загальний фреймворк для тестувальників, щоб вони могли переконатися, що всі аспекти тестування на проникнення ретельно розглянуті. Цей стандарт розроблено для того, щоб надати методологію тестування, яка дозволить визначити найефективніший спосіб проведення тестів, враховуючи потреби конкретної організації. PTES може бути використаний як стандартний чекліст або як складова частина великої методології тестування [37].

PTES має наступні переваги [37]:

- PTES охоплює всі аспекти тестування на проникнення, від збору інформації до післяексплуатаційного етапу. Це гарантує, що жоден аспект не залишиться непротестованим під час ваших випробувань;
- наявні методи стандартизації підходів: завдяки конкретній підготовці для всіх видів випробувань, методи тестування PTES стандартизують свої підходи. Це ускладнює порівняння результатів між різними тестами і спрощує повторення тестів у майбутньому;
- PTES доступний онлайн безкоштовно, і кожен може мати до нього доступ. Це робить його важливим джерелом для будь-якого, хто цікавиться виконанням тестування на проникнення.

2.2.5 Information Systems Security Assessment Framework (ISSAF)

Структура оцінки безпеки інформаційних систем (ISSAF) - це спеціалізований підхід до тестування на проникнення (Група безпеки відкритих інформаційних систем, 2006). Його обширний довідник, який налічує понад 1200 сторінок, розкриває

основи цієї методології тестування. Простий підхід ISSAF легко адаптується для потреб конкретних організацій та індивідуальних тестувальників, дозволяючи створювати персоналізовані плани тестування. Будь-який тестувальник на проникнення, що використовує різні інструменти, повинен дотримуватися методології ISSAF [39].

Важливо зазначити, що ISSAF виходить за межі простого тестування на проникнення: він також включає створення інструментів, які можна використовувати для навчання інших осіб, що мають доступ до мережі. Це також забезпечує, що користувачі мережі дотримуються відповідних правових норм [39, 40].

ISSAF ділить тестування на проникнення на три фази [39]:

- планування та підготовка;
- оцінка;
- звітність, прибирання та знищення.

ISSAF Фаза I: Планування та Підготовка.

Ця фаза коротка і лише описує кроки для обміну початковою інформацією, планування та підготовки тестування. Вона наголошує на необхідності підписання формальної угоди про оцінку перед початком будь-якого тестування. Угода надає основу для цього завдання та взаємного правового захисту, і визначає [39]:

- команди, що беруть участь в оцінці;
- точні дати та часи;
- шляхи ескалації;
- будь-які інші умови.

Завдання фази [39]:

- визначення каналів зв’язку між компанією та командою для тестування на проникнення;
- підтвердження обсягу, підходу та методології;
- узгодження конкретних тестових випадків та шляхів ескалації.

ISSAF Фаза II: Оцінка.

Ця фаза є більш корисною - вона досить детальна і навіть описує деякі інструменти для тестування на проникнення. Цілі описуються як мережі, хости,

додатки та бази даних. До певної міри вона застаріла і не є повною. Ви можете використовувати її як вихідний пункт для фаз оцінки у тестуванні на проникнення, але не для керування всією структурою тестування на проникнення [39].

ISSAF описує окремі кроки оцінки як "шари" тестування на проникнення [39]:

- збір інформації - використовуйте технічні та не технічні методи для отримання відповідної інформації про ціль;
- визначення структури мережі - ідентифікуйте всі системи та ресурси в межах цільової мережі;
- виявлення вразливостей - виявляти вразливості в цілях;
- проникнення;
- отримання доступу та підвищення привілеїв - отримайте привілеї рівня адміністратора на цільовій системі (займіть кореневу директорію);
- додаткове перерахування - отримайте додаткову інформацію про процеси на системах з метою експлуатації мережі/систем (переміщення в бічному напрямку);
- компрометація віддалених користувачів/сайтів - експлуатація відносин довіри та комунікації між віддаленими користувачами та корпоративними мережами;
- збереження доступу - використовуйте приховані канали, backdoors та rootkits для приховування присутності хакера та забезпечення постійного доступу до системи;
- загортання слідів.

ISSAF Фаза III: Звіт.

У цій фазі розглядаються канали зв’язку та типи звітів у проекті. Запропоновано два способи звітування: усний і письмовий [39].

Усний звіт призначений лише для критичних та/або невідкладних питань. Усний зв’язок слід використовувати у випадках, коли виявлені проблеми, які потребують негайної уваги та дій. Прикладом може бути виявлення під час тестування на проникнення того, що система є вразливою та була (або в даний час є) компрометованою. Особливий випадок тут - якщо ми виявимо будь-які незаконні дії в мережі та системах - у таких випадках нам може доведеться зв’язатися з правовими органами до того, як (або навіть без) повідомлення нашому клієнту [39].

Письмовий звіт є формальним результатом тестування на проникнення. Він може мати різні версії для різних зацікавлених осіб у організації. Також він може містити інформацію про проблеми, які вже були обговорені в усному звіті [39].

Згідно з ISSAF, письмовий звіт зазвичай містить [39]:

- огляд для керівництва;
- обсяг проекту;
- використані інструменти тестування на проникнення;
- використані експлойти;
- дата/час тестування;
- усі виводи інструментів та експлойтів;
- список виявлених вразливостей;
- рекомендації з мінімізації виявлених вразливостей, відсортованих за пріоритетом.

ISSAF Фаза IV: Видалення та знищення.

Ця остання частина структури досить коротка і зосереджена на видаленні будь-яких артефактів, що залишилися від тестування на проникнення. Вона дозволяє тестувальникові вибирати спосіб шифрування, очищення та знищення даних, створених під час тестування на проникнення [39].

2.2.6 PCI Penetration Testing Guide

PCI DSS Penetration Testing є типом етичного хакінгу, який симулює мережу та її цільові системи. Тестування на проникнення виходить за межі запуску автоматизованого сканера вразливостей; фахівці з безпеки проводять тести та глибоко проникають у систему [41].

Проведення тестування на проникнення PCI DSS на мережах безпеки, публічних пристроях, програмах, базах даних та інших структурах, що зберігають, обробляють або розповсюджують дані власників карток, означає, що ви намагаєтеся виявити вразливості перед тим, як їх виявлять кіберзлочинці [41].

Якщо компанія використовує вебдодатки для зберігання, обробки або передачі конфіденційної інформації, вони можуть бути вразливими. Багато хакерів можуть компрометувати безпеку підприємств з вебдодатками, які є основою їх бізнесу. Виявлення та усунення вразливостей, які можуть мати вебдодатки, є критичним для бізнесу [41].

Тестування на проникнення для PCI DSS спеціально спрямоване на оцінку безпеки середовища обробки даних власників карток (CDE) організації та забезпечення відповідності вимогам PCI DSS. Основні цілі тестування на проникнення PCI включають [42]:

- виявлення вразливостей, які можна використати: виявлення слабкостей безпеки в межах CDE, які можуть бути використані для незаконного доступу, компрометації систем або витоку чутливих платіжних даних;
- підтвердження контролю безпеки: оцінка ефективності впроваджених заходів безпеки та перевірка їх роботи відповідно до призначення для захисту даних власників карток;
- збереження відповідності;
- пріоритизація ризиків: оцінка серйозності виявлених вразливостей та встановлення пріоритетів для зусиль щодо їх усунення на основі потенційного впливу на безпеку організації;
- постійне удосконалення: регулярна оцінка CDE для попередження еволюції загроз та адаптації стратегій безпеки для ефективного захисту від майбутніх атак.

Тестування на проникнення PCI DSS складається з 5 кроків:

- визначення обсягу: тестувальник налаштує вимоги оцінки відповідності PCI DSS для внутрішньої мережі замовника, щоб визначити обсяг тестування перед початком;
- виявлення: тестувальник виявляє активи вашої мережі в межах визначеного обсягу CDE;
- оцінка: використовуючи деталі, знайдені на першому кроці, тестується мережа та додатки на наявність можливих вразливостей безпеки;

- звітування: тестувальник комплексно оцінює результати тестування, готує повний звіт, пояснюючи методологію та результати, і надає чітку інформацію про всі етапи тестування для представлення доказів назначеному QSA або іншим зацікавленим сторонам;
- повторний тест: процеси перевіряються знову, щоб переконатися, що всі виявлені проблеми були успішно вирішені.

На відміну від реального зловмисника, тестування на проникнення проводиться декілька годин на тестування конкретного середовища. Тому клієнт, повинен вирішити, де потрібно витратити більшість часу тестувальника на проникнення. До початку тестування якість та обсяг наданої інформації аналітику має великий вплив на час, необхідний для тестування [42].

Методологія тестування на проникнення поділена на три типи тестів: чорна скринька, біла скринька та сіра скринька. У тестах чорної скриньки експерт, який виконує тест, не має жодної інформації про ваше середовище перед тестом [42].

У тестах білої скриньки експерт отримує детальну інформацію про ваше середовище перед тестами. У тестах на проникнення сірої скриньки експерт має обмежену інформацію про ваше середовище перед тестами [42].

Замовник повинен визначити тип тестування на проникнення відповідно до тих областей, на які ви будете зосереджуватися. Також існує багато різних типів тестів, що включають тестування на проникнення [42].

PCI penetration тестування та звичайне тестування на проникнення мають багато подібностей у методології та цілях, але вони мають важливі відмінності, які потрібно враховувати. Хоча обидва типи тестів спрямовані на виявлення проблем і покращення безпеки організації, основна різниця полягає у обсязі та акценті тестів, а також обов’язковій частоті проведення тестування [42].

У тестуванні на проникнення PCI DSS використовується широкий спектр інструментів для оцінки безпеки середовища та виявлення потенційних вразливостей, які можуть бути використані зловмисниками [42].

Ці інструменти відіграють важливу роль у оцінці різних аспектів безпеки організації, включаючи конфігурації мережі, вебзастосунки, бездротові мережі та

вразливості систем. Вибір інструментів може варіюватися залежно від конкретних потреб і цілей тестування на проникнення, але деякі широко використовувані інструменти в тестуванні на проникнення PCI включають [42]:

Nmap: Для сканування мережі та виявлення відкритих портів і служб, а також перевірки сегментації.

Nessus: Для автоматизації процесу виявлення відомих вразливостей у системах, мережах та додатках. Він також може використовуватися для сканування вразливостей PCI, оскільки може проводити сканування вразливостей PCI.

Metasploit: Комплексний фреймворк для розробки та виконання використання коду проти виявлених вразливостей.

Burp Suite: Потужний інструмент для тестування безпеки вебзастосунків з такими функціями, як перехоплення проксі, сканування вразливостей вебзастосунків та маніпулювання сесіями.

Wireshark: Аналізатор протоколів мережі для захоплення та аналізу мережевого трафіку, що надає цінну інформацію про потенційні вразливості та вектори атак. Особливо корисно для перевірки того, чи передається карта даних по мережі без шифрування.

Інструмент для скрапінгу пам’яті: Інструмент на основі регулярних виразів для скрапінгу оперативної пам’яті може бути корисним у тестуванні безпеки PCI для виявлення даних кредитних карток, тимчасово збережених в пам’яті сервера, розташованого в середовищі, після їхнього компрометування тестувальником на проникнення [42].

2.2.7 NIST Special Publication 800-115

Національний інститут стандартів та технологій, або NIST, є організацією, що входить до складу Міністерства торгівлі США і має за мету бути лідером у сфері інновацій та технологій, надаючи справедливі стандарти та рішення [43].

Основними компетенціями NIST є точне відстеження, розробка та використання стандартів. Ці компетенції впливають на надійність інформації, яку

надає організація. Як лідер у галузі, NIST має можливість надавати якісні рішення, які можуть бути використані організаціями для розробки безпечних практик інформаційної безпеки та проведення випробувань на стійкість [43, 44].

NIST публікує документи, які можуть бути корисними для розробки стратегій і методик, які використовуються фахівцями з інформаційної безпеки. Один із цих документів - NIST SP 800-115, Технічний посібник з тестування та оцінки інформаційної безпеки, який використовується при плануванні та впровадженні ефективних процесів та процедур забезпечення безпеки [43].

У сфері тестування на проникнення NIST SP 800-115 є цінним джерелом, що впливає на методології, які використовуються тестувальниками для ідентифікації вразливостей в організаціях [43].

Оцінка безпеки, тестування та перевірка безпеки є важливими з двох основних причин [43]:

- перевірка, чи працюють впроваджені заходи безпеки так, як очікувалося;
- виявлення нових слабких місць у безпеці.

Незважаючи на важливість цих процесів, також важливо, наскільки добре ви їх виконуєте. Щоб допомогти з цим, Національний інститут стандартів та технологій (NIST) у Спеціальних Публікаціях 800-115 надає технічні рекомендації для тестування та оцінки безпеки [43].

NIST 800-115 розкладено на кілька розділів, які охоплюють різні аспекти тестування безпеки [43]:

- огляд тестування та перевірки безпеки;
- техніки огляду;
- техніки ідентифікації та аналізу цілей;
- техніки перевірки вразливостей цілей;
- планування оцінки безпеки;
- виконання оцінки безпеки;
- пост-тестувальні дії.

Розділ «Огляд тестування та екзаменації безпеки». Цей розділ закладає основу для тестування безпеки та планування. Згідно з NIST SP 800-115, оцінка безпеки повинна складатися принаймні з таких етапів [43]:

- планування;
- виконання;
- пост-виконання.

Стандарт також визначає 3 типи методів оцінки (Тестування: Порівняння фактичної поведінки з очікуваною поведінкою; Екзаменація: Перевірка, огляд, рецензування, спостереження, вивчення або аналіз об'єкта для поліпшення розуміння його; Інтерв'ю: Обговорення з працівниками організації в групах або індивідуально для отримання пояснень) [43].

Розділ «Техніки огляду». У цьому розділі розглядаються різноманітні техніки огляду, такі як перегляд документації, журналів, наборів правил та конфігурацій. Крім того, згадується перехоплення мережі, яке може бути використане для ідентифікації та аналізу цілей. І, нарешті, йдеться про перевірку цілісності файлів для перевірки, чи не було змінено жодних системних файлів або критичних файлів [43].

Розділ «Техніки ідентифікації та аналізу цілей». У цьому розділі розглядається ідентифікація портів, служб та систем у мережі. Наступним кроком є виявлення будь-яких вразливостей в їх роботі. Техніки, що розглядаються в цьому розділі [43]:

- виявлення мережі;
- ідентифікація портів та служб у мережі;
- сканування вразливостей;
- бездротове сканування (пасивне та активне сканування, відстеження місцезнаходження бездротових пристроїв, сканування Bluetooth).

Розділ «Техніки перевірки вразливостей цілей». У цьому розділі розглядається підтвердження існування вразливості та розуміння наслідків її використання. Він охоплює технічні слабкості, а також слабкості, пов'язані з відсутністю навчання: взлом пароля; тестування на проникнення; соціальна інженерія [43].

Розділ «Планування оцінки безпеки». Якщо ви не належним чином плануєте оцінку безпеки, то можете витратити свої ресурси даремно і так і не досягти того, що

мали на увазі. Цей розділ присвячений плануванню процесу оцінки безпеки. Він надає поради щодо [43]:

- розроблення політики оцінки безпеки;
- підставки і планування оцінок;
- вибір та налаштування методик;
- логістика оцінки (вибір та навички оцінювачів, місце проведення, вибір інструментів та ресурсів);
- розробка плану оцінки;
- вирішення правових питань.

Розділ «Виконання оцінки безпеки». Виконання - це те, що відбувається після планування і має важливе значення для ефективного виконання плану оцінки. Якщо є причина відхилитися від плану, ситуація повинна бути переглянута для прийняття рішення. Цей розділ надає керівні принципи для плавного проведення оцінки безпеки і включає: координацію, оцінку, аналіз та обробку даних [43].

Розділ «Пост-тестувальні дії». Як і вказує назва, це те, що відбувається після тестування. На цій фазі зібрані дані перетворюються на точки дії. Пост-тестувальні заходи спрямовані на збір результатів з попереднього розділу та створення плану для виправлення виявлених вразливостей. NIST надає керівні принципи для наступних пост-тестувальних заходів [43]:

- рекомендації з виправлення;
- звітність;
- виправлення/зменшення ризиків.

Щоб максимально використовувати техніки, вказані в NIST SP 800-15, необхідно мати достатньо навчений персонал. NIST також згадує певні базові набори навичок для кожної з цих технік, якими можна користуватися [43].

2.2.8 MITRE ATT&CK

MITRE ATT&CK - це база знань про тактики та методи діяльності зловмисників, побудована на основі реальних спостережень. Вона використовується

для розробки моделей та методологій протидії загрозам для компаній у різних галузях економіки та у спільноті розробників кібербезпеки [45].

ATT&CK є відкритою і безкоштовною для використання всіма організаціями та індивідуальними користувачами. За даними MITRE, на сьогодні база містить понад 2900 тактик, технік і процедур, які використовують зловмисники для атак [45].

Інструмент став невід'ємною частиною аналізу загроз та захисту інформації для багатьох компаній та установ. Однак, через розмаїття та обширність ATT&CK може виникати складність у визначенні оптимального шляху для реалізації цієї концепції [45, 48].

У фреймворк входить близько 180 методів і 375 підметодів. Вони представляють різноманітні способи, які використовуються зловмисниками для вчинення шкідливих дій. Ця велика кількість деталей може ускладнити прийняття рішення про те, на яких аспектах варто зосередити увагу (рис. 2.15) [46,47].

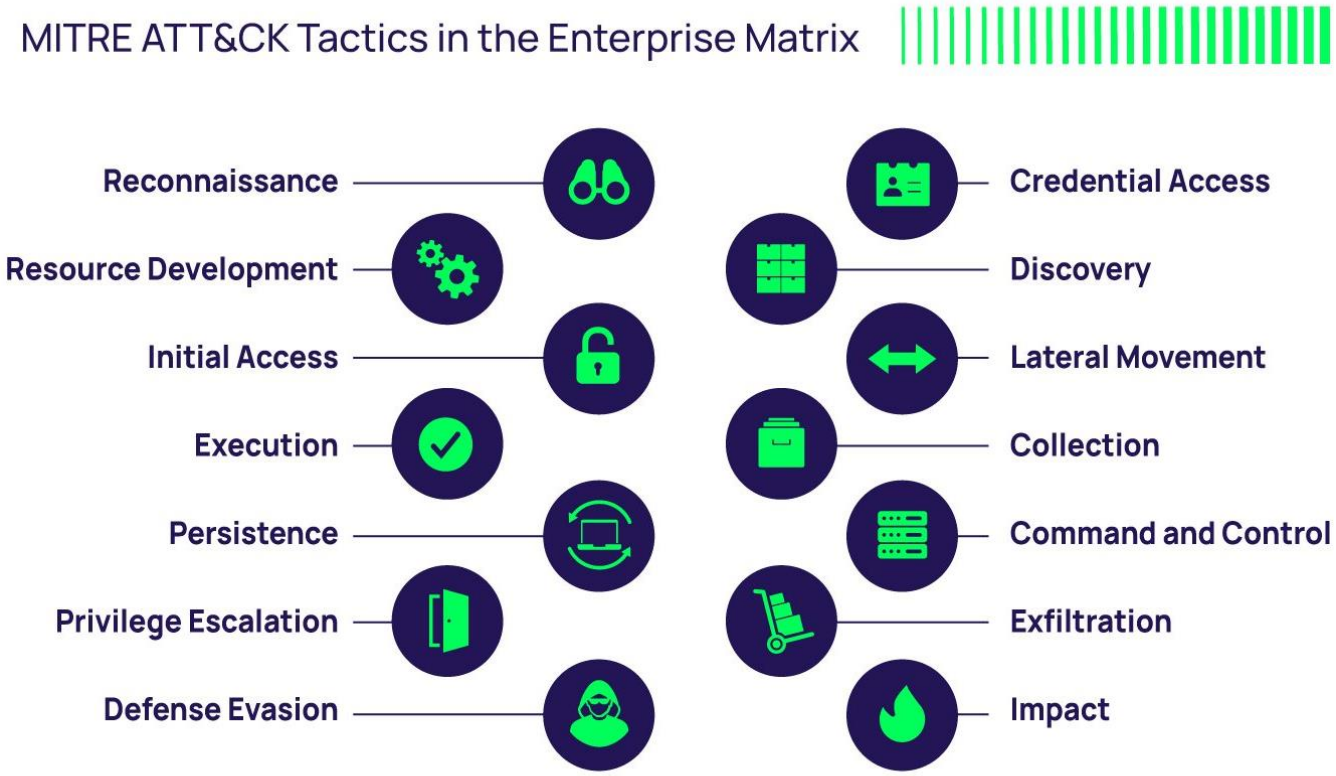


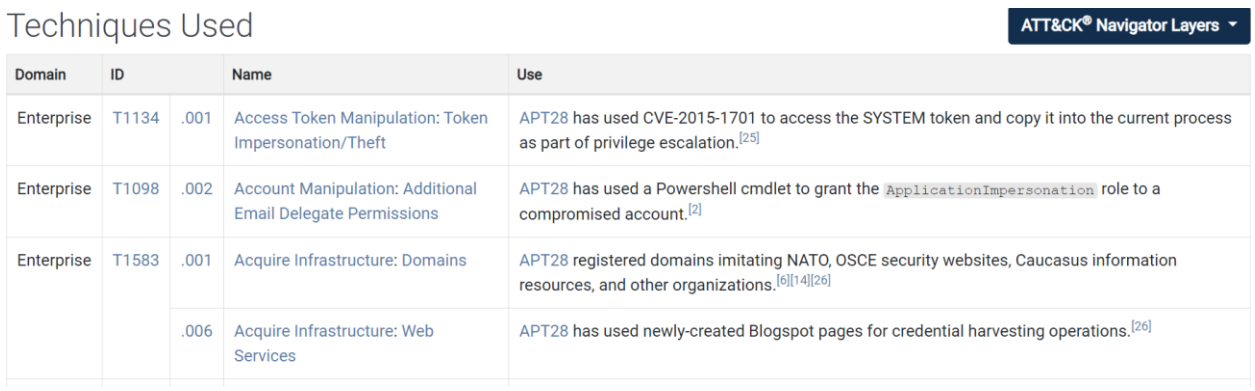
Рисунок 2.15 – Матриця MITRE ATT&CK

База MITRE ATT&CK містить перелік найбільших хакерських угрупувань в світі. На разі актуально досліджувати хакерські угрупування з рф, а саме APT28 та APT29. Це одні з найбільших угрупувань, які цілодобово здійснюють кібератаки на різну інфраструктуру в Україні [49].

APT28 - хакерське угрупування, яку приписують Головному розвідувальному управлінню (ГРУ) 85-го Головного центру спеціального обслуговування (ГЦСЗ) в/ч 26165. Це угруповання діє з 2004 року (рис. 2.16) [49].

APT29 - хакерське угрупування, яку приписують Службі зовнішньої розвідки Росії (СВР). Вони діють з 2008 року, часто націлюючись на урядові мережі Європи та країнах НАТО, дослідницькі інститути та аналітичні центри. Повідомляється, що APT29 скомпрометував Національний комітет Демократичної партії, починаючи з літа 2015 року (рис. 2.17) [50].

Фреймворк зазначає, як ті чи інші користувачі даних угрупувань використовують різні техніки та тактики для здійснення зловмисних дій.

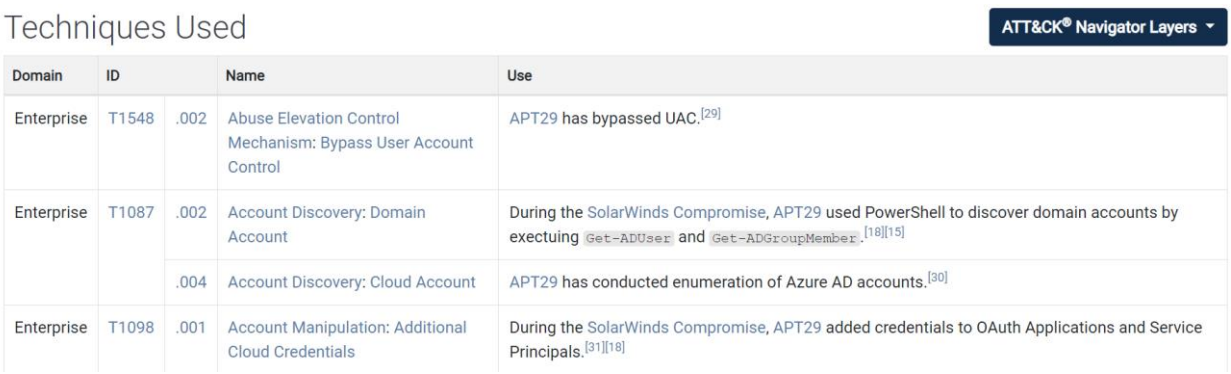


Techniques Used

ATT&CK® Navigator Layers

Domain	ID		Name	Use
Enterprise	T1134	.001	Access Token Manipulation: Token Impersonation/Theft	APT28 has used CVE-2015-1701 to access the SYSTEM token and copy it into the current process as part of privilege escalation.[25]
Enterprise	T1098	.002	Account Manipulation: Additional Email Delegate Permissions	APT28 has used a Powershell cmdlet to grant the `ApplicationImpersonation` role to a compromised account.[2]
Enterprise	T1583	.001	Acquire Infrastructure: Domains	APT28 registered domains imitating NATO, OSCE security websites, Caucasus information resources, and other organizations.[6][14][26]
		.006	Acquire Infrastructure: Web Services	APT28 has used newly-created Blogspot pages for credential harvesting operations.[26]

Рисунок 2.16 – Використання технік угрупуванням APT28



Techniques Used

ATT&CK® Navigator Layers

Domain	ID		Name	Use
Enterprise	T1548	.002	Abuse Elevation Control Mechanism: Bypass User Account Control	APT29 has bypassed UAC.[29]
Enterprise	T1087	.002	Account Discovery: Domain Account	During the SolarWinds Compromise, APT29 used PowerShell to discover domain accounts by exectuing `Get-ADUser` and `Get-ADGroupMember`.[18][15]
		.004	Account Discovery: Cloud Account	APT29 has conducted enumeration of Azure AD accounts.[30]
Enterprise	T1098	.001	Account Manipulation: Additional Cloud Credentials	During the SolarWinds Compromise, APT29 added credentials to OAuth Applications and Service Principals.[31][18]

Рисунок 2.17 – Використання технік угрупуванням APT29

Окрім різних технік та тактик фреймворк MITRE пропонує різні рішення для запобігання тих чи інших кібератак, що значно полегшує роботи компаній, які намагаються створити потужну кібер-мережу.

Висновки за розділом 2

У другому розділі дипломної роботи було досліджено загальну концепцію та основні принципи аналізу вебексплойтів, а також проаналізовано різні методології, які використовуються для виявлення вразливостей в вебдодатках. Згідно з дослідженнями, вебексплойти становлять серйозну загрозу для безпеки інформації та приватності користувачів, оскільки вони можуть бути використані для несанкціонованого доступу до системи або конфіденційної інформації.

Під час аналізу основних принципів аналізу вебексплойтів було виявлено, що процес включає в себе виявлення, аналіз та використання вразливостей. Для цього використовуються різні методи, такі як зовнішнє сканування мережі, аналіз даних з лог-файлів, а також проведення тестування на проникнення.

Крім того, досліджено різні методології аналізу вебдодатків на вразливості, включаючи ті, що базуються на зовнішньому скануванні, а також внутрішній аналіз програмного коду. Ці методології дозволяють ідентифікувати потенційні недоліки та слабкі місця в програмному забезпеченні, що є критичним для забезпечення безпеки вебдодатків.

РОЗДІЛ 3
РОЗРОБКА МОДЕЛІ АНАЛІЗУ ВЕБЕКСПЛОЙТІВ

3.1 Встановлення середовища для тестування

Було обрано середовище для тестування - CMS Wordpress, так як, через велику кількість користувачів, WordPress стає привабливою мішенню для зловмисників, оскільки потенційної шкоди можна завдати багатьом вебсайтам одночасно. Також, CMS WordPress містить безліч плагінів і тем, що розширюють функціонал системи.

Розробники плагінів та тем не завжди дотримуються найвищих стандартів безпеки, що може призводити до появи вразливостей в вебдодатку. Крім того, активний розвиток WordPress і його постійні оновлення можуть приховувати недоліки безпеки, які тестувальники можуть виявити перед релізом нових версій. І, нарешті, широкий функціонал платформи, такий як можливість створювати власні додаткові поля або налаштовувати права доступу, створює додаткові можливості для потенційних вразливостей.

Було встановлено Wordpress на Ubuntu 22.04. Для тестування було встановлено вразливу тему для Wordpress (рис.3.1) з github [https://github.com/vavkamil/dvwp].

Активні та неактивні плагіни наведено на рис. 3.2 - 3.3.

До плагінів належать:

- Infinite WP client - версія 1.9.4.4;
- Social Warfare - версія 3.5.2;
- Wordpress File Upload - версія 4.12.2;
- WP-Advanced-Search - версія 3.3.3;
- Askimet Anti-Spam - версія 4.1.3;
- Backup and Staging by WP Time Capsule - версія 1.21.15;
- Hello Dolly - версія 1.7.2.

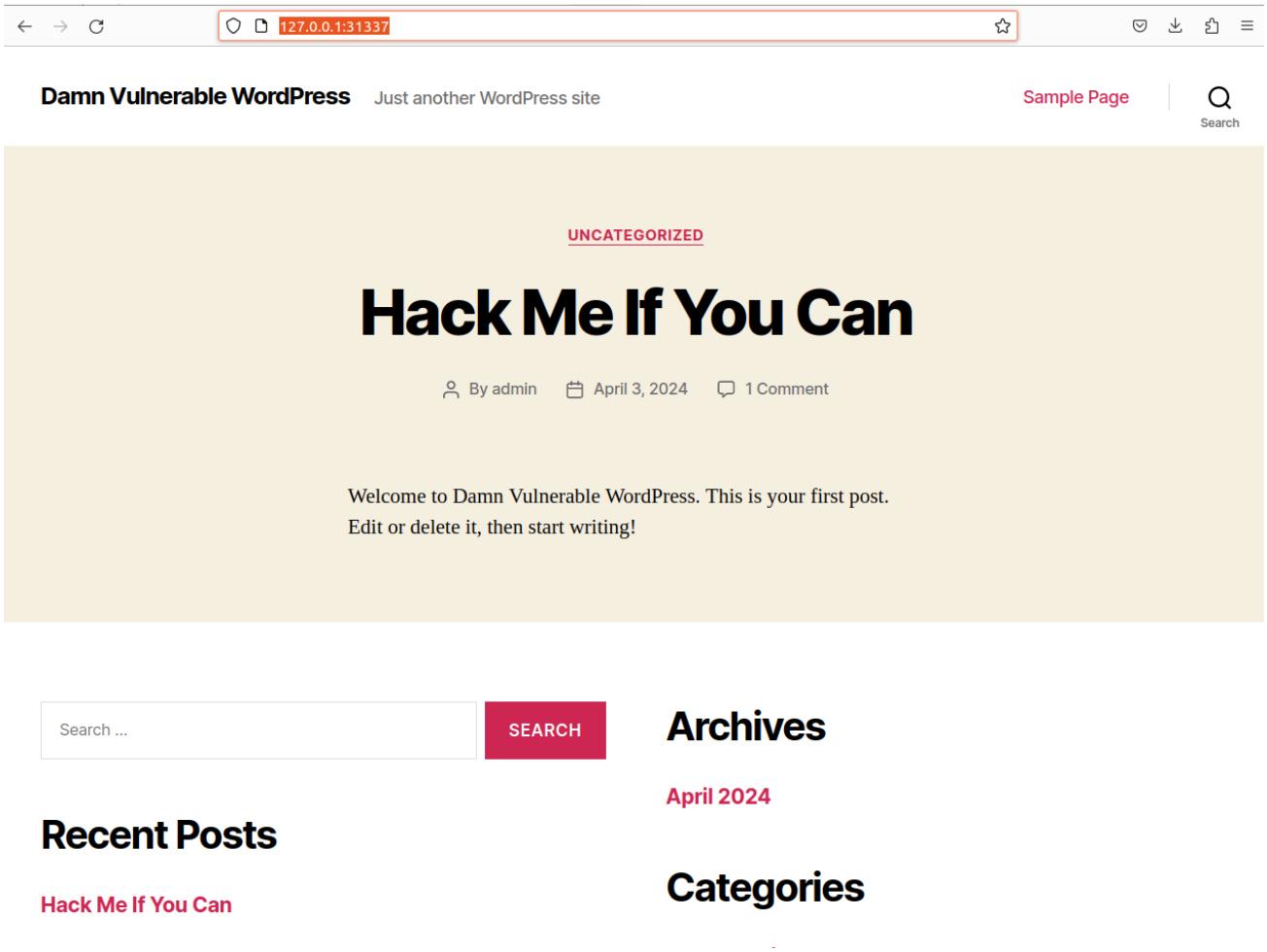


Рисунок 3.1 – Встановлений Wordpress

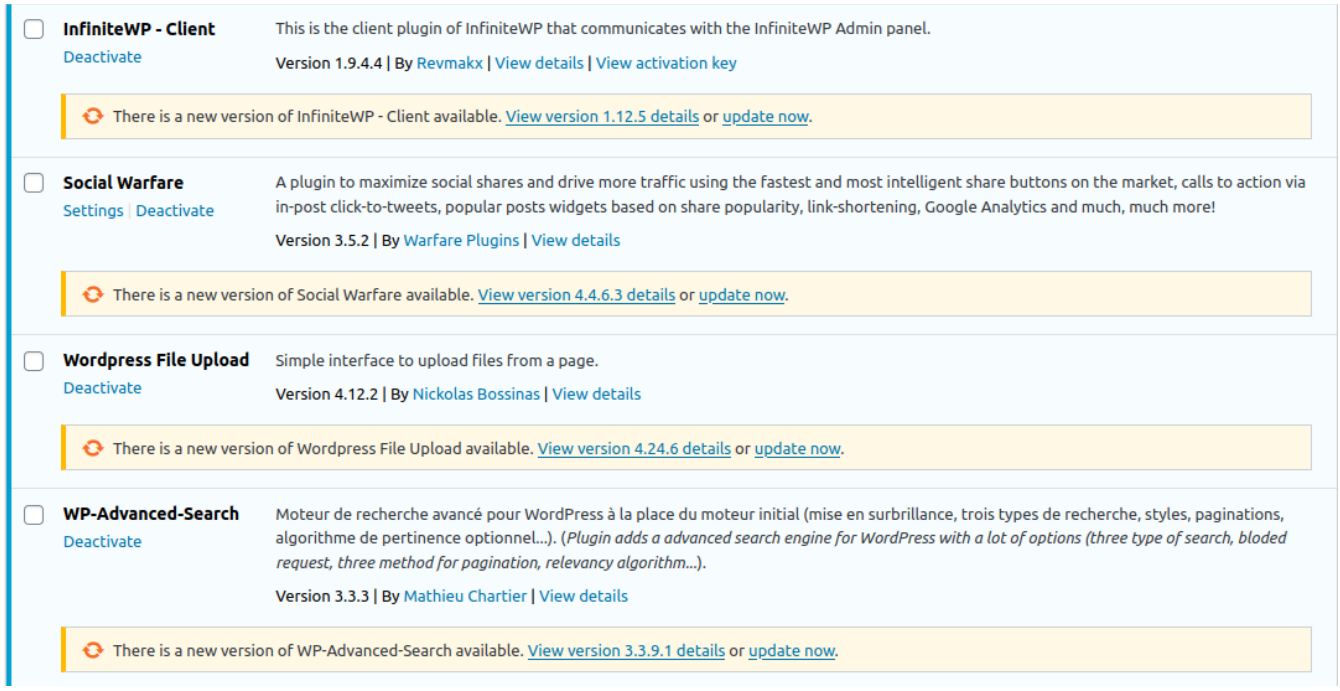


Рисунок 3.2 – Активні плагіни

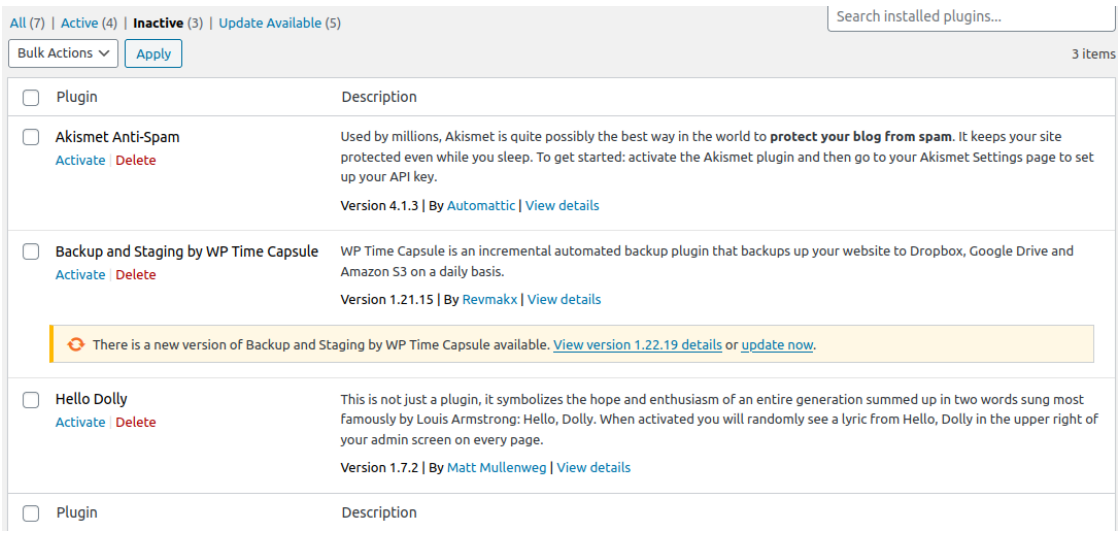


Рисунок 3.3 – Неактивні плагіни

У вебдодатку також присутні поля пошуку (рис.3.4), які можуть бути вразливі. Пошук реалізований за допомогою плагіну WP-Advanced-Search.

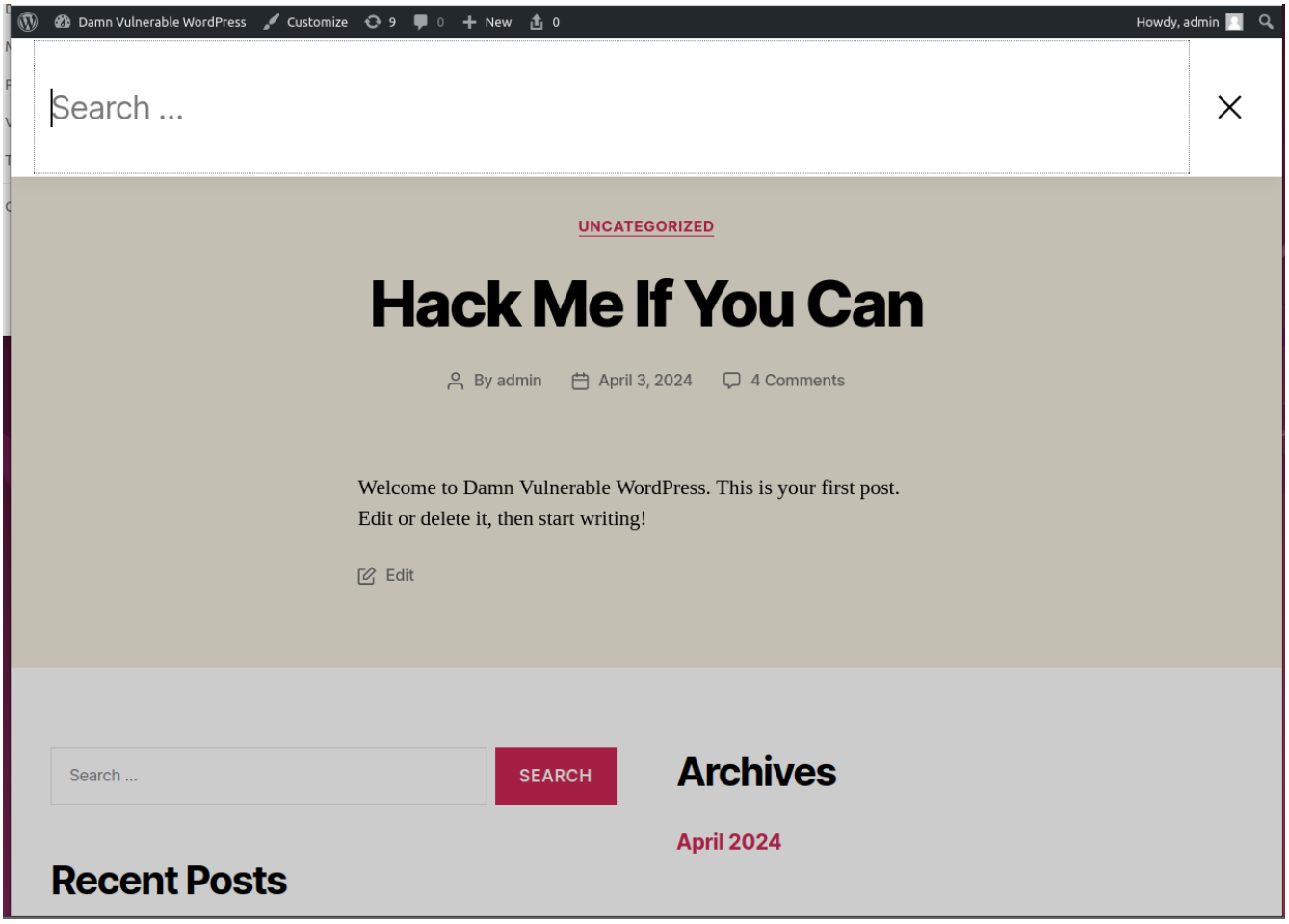


Рисунок 3.4 – Поля пошуку

3.2 Формування моделі аналізу

Wordpress містить безліч плагінів для аналізу вразливостей у вебдодатку, проте ці плагіни можуть і самі бути вразливими.

Коли мова йде про аналіз вразливостей у вебдодатках, створення власного скрипту може бути кращим рішенням, ніж використання плагінів у WordPress, тому що власний скрипт може бути розроблений під ваш вебдодаток, що дозволить врахувати особливості даного вебдодатку. Також при розробці і використанні власного рішення, ви можете мати повний контроль над кодом, що дозволяє гарантувати безпеку вашого рішення, оскільки ви не залежите від сторонніх плагінів та бібліотек і використовуєте свої функції. Використовуючи власний скрипт, можна оперативно виправляти вразливості, які можуть бути знайдені.

Отже, власний скрипт для аналізу вразливостей може бути більш надійним та ефективним варіантом, ніж використання плагінів у WordPress.

Модель аналізу вебдодатку на CMS Wordpress має містити такі етапи:

1. Початкова розвідка вебдодатку та сканування всіх наявних сервісів, портів та служб. Використовуючи інструменти, такі як nmap, скрипт зможе сканувати порти та визначати відкриті служби.

2. Аналіз вебсайту на вразливості. Скрипт зможе використовувати інструменти, такі як Nikto або OWASP ZAP, для сканування вебсайту на наявність відомих вразливостей.

3. Генерація звіту. Скрипт зможе створювати звіт з виявленими вразливостями та рекомендаціями щодо їх виправлення.

На першому етапі моделі використаємо інструмент nmap та dirb для пошуку файлів/директорій, до яких має доступ звичайний користувач.

Nmap - це спеціальна утиліта, яка має відкритий вихідний код і дозволяє здійснювати перевірку мереж та безпеки в цих мережах. Додаток формує звіт з даними, які допомагають досліджувати мережу. Використовуючи спеціальні флаги утиліта здатна виводити потрібну інформацію, що значно полегшує користування утилітою.

Для прикладу, під час дослідження безпеки вебдодатку було використано наступні флаги:

-p - дозволяє вказати потрібний діапазон портів для сканування мережі;

-sV - дозволяє вивчати відкриті порти та запущені служби;

-sC - виконує сканування сценаріїв за допомогою стандартного набору сценаріїв, так як за замовченням, не всі сценарії можуть запускатись для сканування мережі;

-A - активує функції визначення ОС та їх версій.

Таким чином, використовуючи певні флаги утиліта Nmap розширить можливості дослідження мереж.

DIRB - це програмне забезпечення, яке дозволяє взаємодіяти з вмістом вебсторінок. Сканер виконує сканування на приховані файли та директорії, які здатні послабити рівень безпеки додатку.

Дана утиліта працює за допомогою вбудованих словників, які містять список можливих файлів та директорій. Окрім наявгих словників, можна створювати власні словники для власних потреб.

DIRB також дозволяє використовувати флаги, якими можна сортувати потрібну інформацію. Наприклад:

-f - виявлення помилки «NOT FOUND 404»;

-c - використання cookies;

-v - відображати сторінки «NOT FOUND 404»;

-l - відображати заголовки «Location»;

-r - статистика сканування;

-i - використовувати пошук без врахування регістра.

На другому етапі використаємо інструменти для пошуку вразливостей такі як WPScan, nikto, SQLmap, PwnXSS.

WPScan - це сканер, який широко використовується спеціалістами з кібербезпеки, які бажають визначити усі можливі слабкі місця в CMS WordPress. Сканер дозволяє виявляти вразливості в версіях систем, в плагінах, а також темах.

Таким чином, використання даного програмного забезпечення - це спосіб забезпечити надійну безпеку вебдодатку.

Окрім цього, WPScan здатний проводити брутфорс атаку на WordPress. Використовуючи додаткові словники з даними можна підібрати дані для авторизації.

Nikto - програмне забезпечення для пошуку можливих вразливостей у вебдодатках. Це безкоштовний сканер, який виконує сканування використовуючи стратегії сканування методом чорного ящика.

SQLmap - сканер з відкритим вихідним ходом, який призначений для тестування на проникнення. Він дозволяє автоматизувати процес виявлення та експлуатування вразливостей SQL ін’єкцій та захоплення вебдодатків та вебсерверів.

Утиліта PwnXSS виконує пошук вразливостей міжсайтового сценарію XSS. Програмне забезпечення - є безкоштовним, написано на мові програмування python та здатне полегшити дослідження вебдодатків.

На третьому етапі потрібно згенерувати звіт з усіма результатами. Звіт має містити детальний результат тестування вебдодатку за допомогою усіх згаданих вище утиліт та програм.

3.3 Практична реалізація моделі аналізу

У віртуальному середовищі Oracle Virtual Box було створено на налаштовано віртуально ОС Ubuntu 22.04, де було написано bin/bash скрипт для аналізу вебдодатку WordPress, про який було написано раніше. Даний скрипт забезпечує автоматизоване сканування вебдодатку шляхом запуску утиліт таких як NMAP, Dirb, Nikto, SQLMap, WPScan та PwnXSS. Також, всі результати записуються в окремий файл для подальшого вивчення всіх знайдених проблем безпеки вебдодатку (рис. 3.5 - 3.6). На рисунку 3.5 та 3.6 можна побачити, що користувача повідомляють про те, що почалось сканування вебдодатку певним сканером, а також повідомляють про те, що сканування завершилось. Так користувач зможе прослідкувати за прогресом сканування.

```
 GNU nano 6.2                                    script.sh
#!/bin/bash


echo "Start nmap"
echo "_______  _______    ___    _____   _______ "
echo "|            |      / \    |    \      |   "
echo "|            |     /   \   |     \     |   "
echo "|_____       |     ____    |____/      |   "
echo "      |      |    /    \   |    \      |   "
echo "      |      |   /      \  |     \     |   "
echo "______|      |  /        \ |      \    |   "

nmap -p- -sV -sC -A 127.0.0.1 | tee -a "scan-results.txt"
echo "Finished"


echo "Start dirb"
echo "_______  _______    ___    _____   _______ "
echo "|            |      / \    |    \      |   "
echo "|            |     /   \   |     \     |   "
echo "|_____       |     ____    |____/      |   "
echo "      |      |    /    \   |    \      |   "
echo "      |      |   /      \  |     \     |   "
echo "______|      |  /        \ |      \    |   "

dirb http://127.0.0.1:31337/ -f | tee -a "scan-results.txt"
echo "Finished"

echo "Start Nikto"
echo "_______  _______    ___    _____   _______ "
echo "|            |      / \    |    \      |   "
echo "|            |     /   \   |     \     |   "
echo "|_____       |     ____    |____/      |   "
echo "      |      |    /    \   |    \      |   "
echo "      |      |   /      \  |     \     |   "
echo "______|      |  /        \ |      \    |   "

nikto -h http://127.0.0.1:31337/ | tee -a "scan-results.txt"
echo "Finished"


echo "Start SQLmap"
echo "_______  _______    ___    _____   _______ "
echo "|            |      / \    |    \      |   "
echo "|            |     /   \   |     \     |   "
echo "|_____       |     ____    |____/      |   "
echo "      |      |    /    \   |    \      |   "
echo "      |      |   /      \  |     \     |   "
echo "______|      |  /        \ |      \    |   "

sqlmap -u 127.0.0.1:31337/?s=gngh | tee -a "scan-results.txt"
echo "Finished"
```

Рисунок 3.5 – Реалізований скрипт /bin/bash

```
echo "Start WPSCAN"
echo "_______  _______    ___    _____   _______ "
echo "|            |      / \    |    \      |   "
echo "|            |     /   \   |     \     |   "
echo "|_____       |     ____    |____/      |   "
echo "      |      |    /    \   |    \      |   "
echo "      |      |   /      \  |     \     |   "
echo "______|      |  /        \ |      \    |   "

wpscan --url http://127.0.0.1:31337/ --api-token IAph8RXF1kSeypxDAsXDVrmPr3AaFsHXimatV0YeeS4 | tee -a "scan-results.txt"
echo "Finished"

echo "Start PwnXSS"
echo "_______  _______    ___    _____   _______ "
echo "|            |      / \    |    \      |   "
echo "|            |     /   \   |     \     |   "
echo "|_____       |     ____    |____/      |   "
echo "      |      |    /    \   |    \      |   "
echo "      |      |   /      \  |     \     |   "
echo "______|      |  /        \ |      \    |   "

python3 PwnXSS/pwnxss.py -u http://127.0.0.1:31337/ | grep -i "WARNING" | tee -a "scan-results.txt"
echo "Finished"
```

Рисунок 3.6 – Реалізований скрипт /bin/bash

Після запуску скрипта в терміналі Ubuntu 22.04 програма запускає NMAP 7.80. Він дозволяє переглянути всі наявні порти, служби та сервіси (рис. 3.7-3.8).

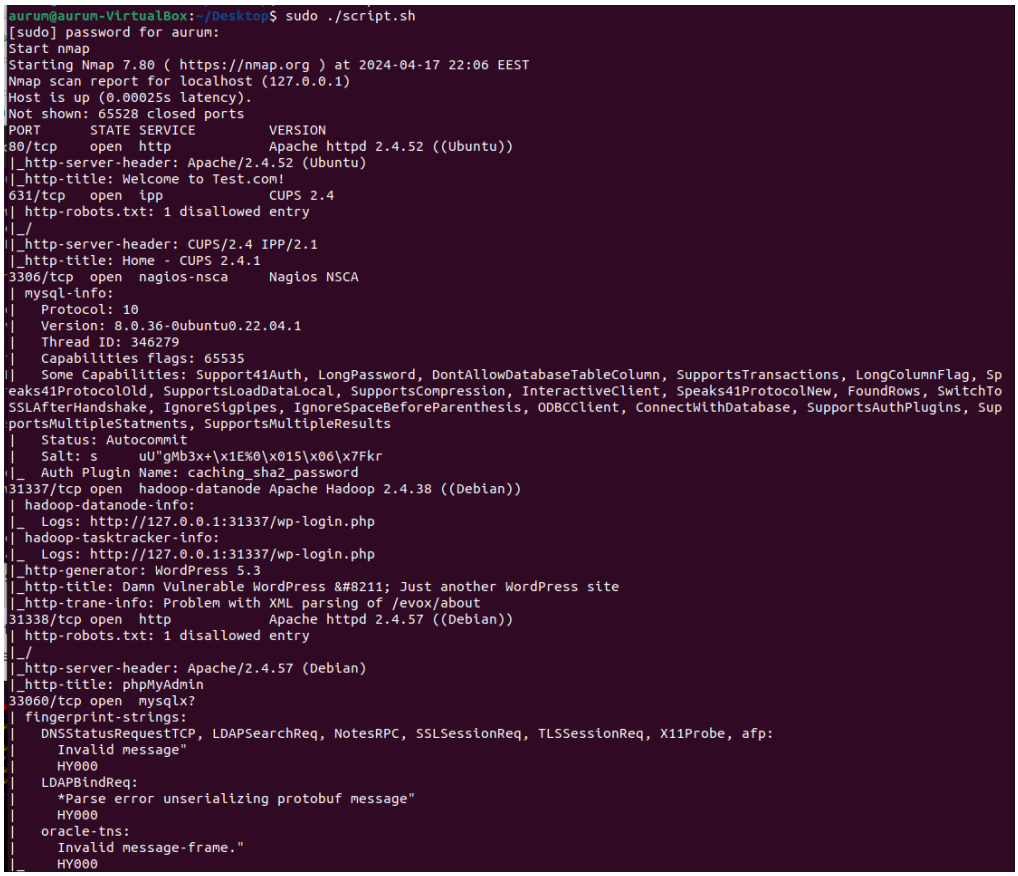


Рисунок 3.7 – Результат сканування Nmap

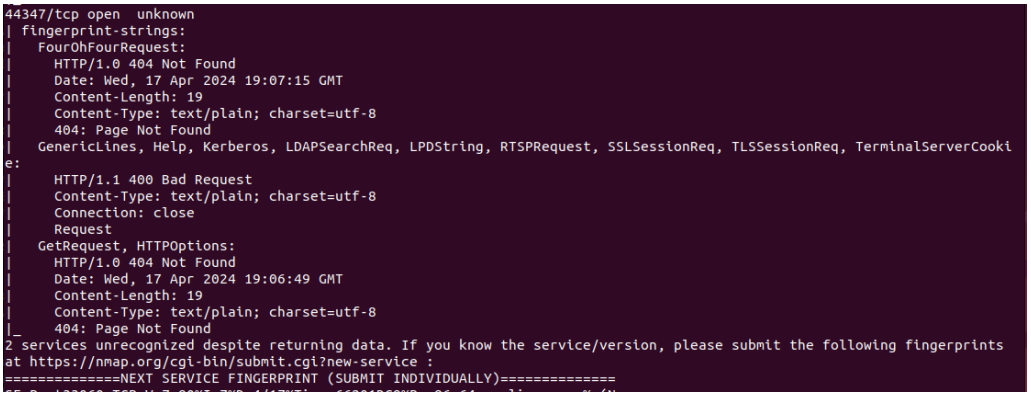


Рисунок 3.8 – Результат сканування Nmap

Наступна утиліта - це DIRB, яка проводить сканування на наявність прихованих директорій та файлів в вебдодатку. На рисунку 3.9 - результат проведення сканування

dirb, який показав, що досліджуваний вебдодаток налічує 7 різних файлів та декілька директорій.

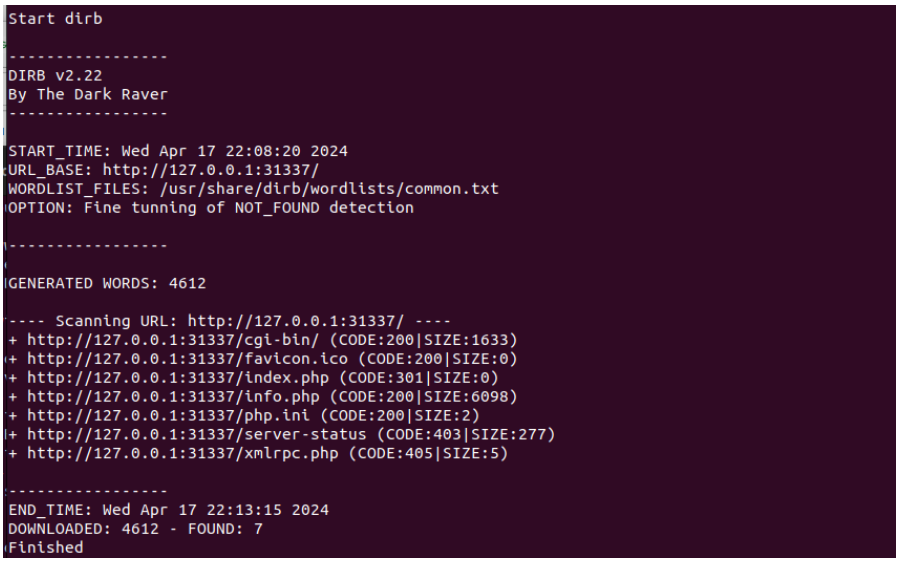


Рисунок 3.9 – Результат сканування dirb

Наступний крок - це сканування за допомогою утиліти Nikto. Даний сканер дозволяє проводити сканування вебдодатку на наявність різних вразливостей. Таким чином, під час сканування вебдодатку утиліта виявила багато можливих вразливих місць, які здатні порушити безпеку додатків (рис. 3.10-3.16).

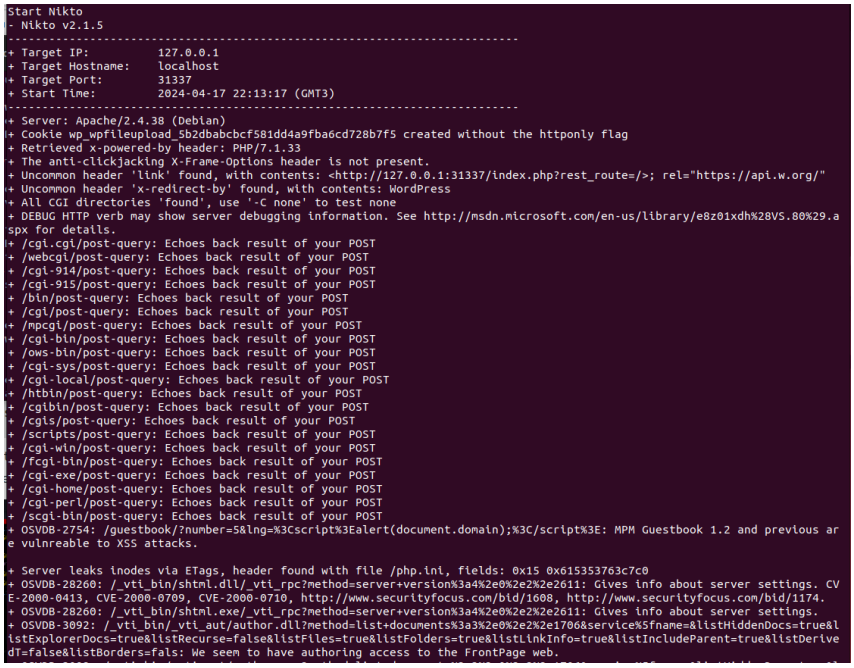


Рисунок 3.10 – Результат сканування Nikto

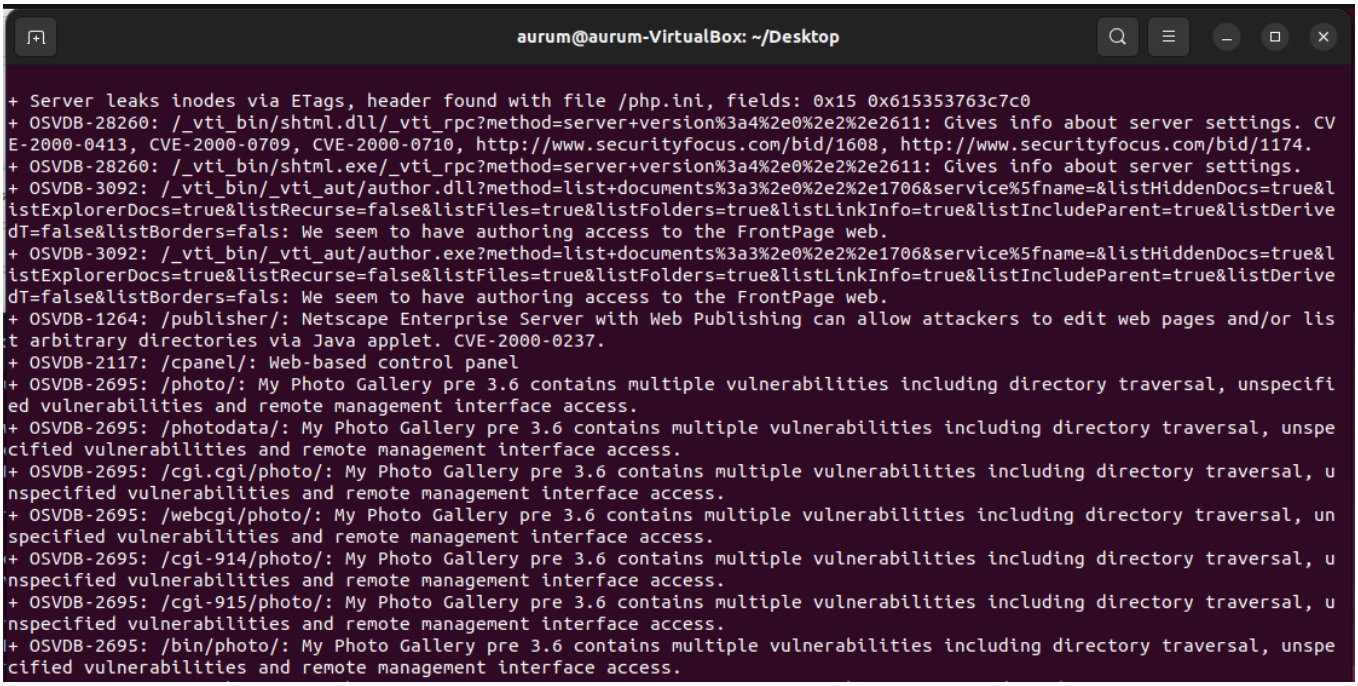


Рисунок 3.11 – Результат сканування Nikto

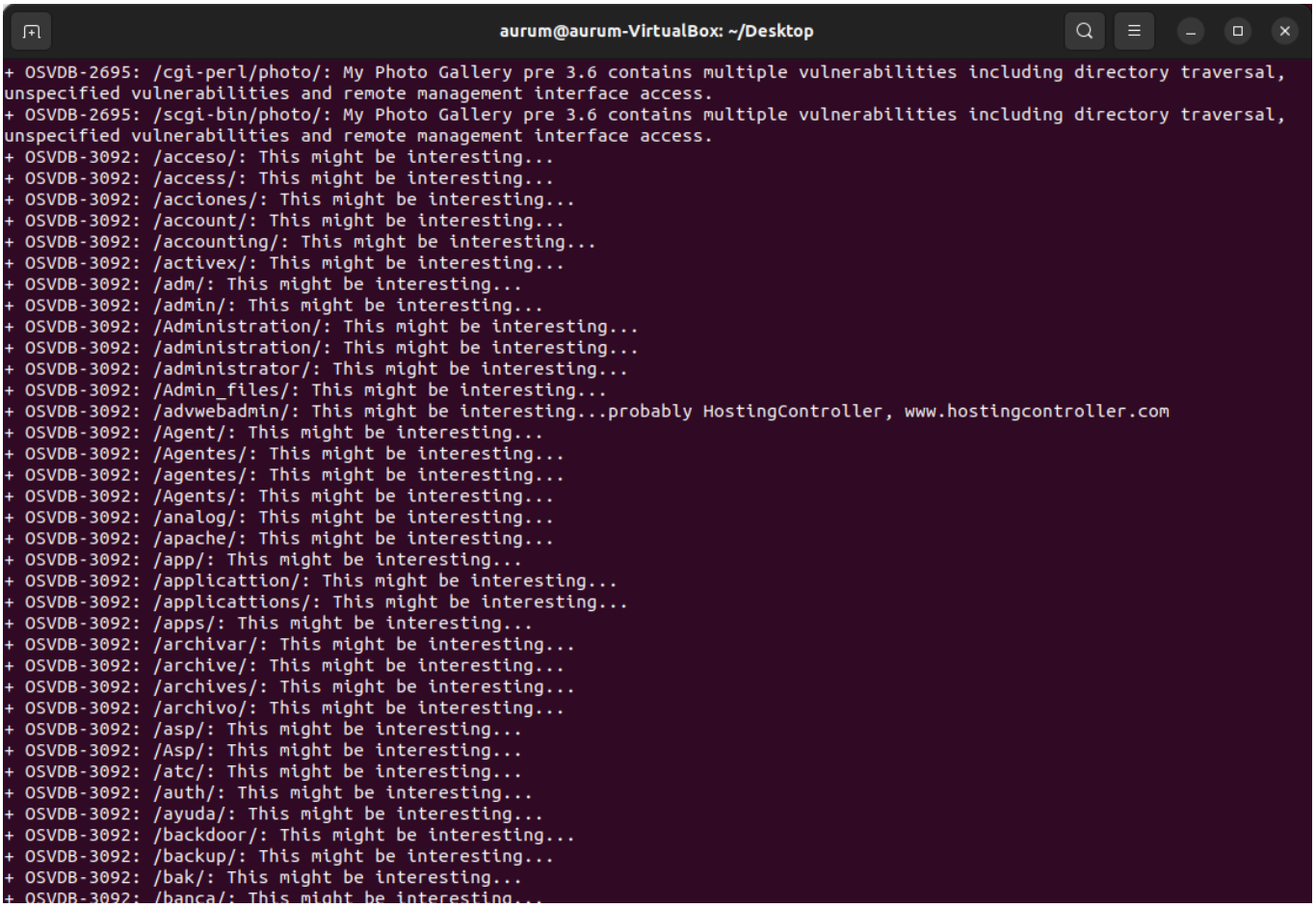


Рисунок 3.12 – Результат сканування Nikto

```
+ OSVDB-3092: /scgi-bin/stats/: This might be interesting...
+ OSVDB-3092: /cgi.cgi/stats_old/: This might be interesting...
+ OSVDB-3092: /webcgi/stats_old/: This might be interesting...
+ OSVDB-3092: /cgi-914/stats_old/: This might be interesting...
+ OSVDB-3092: /cgi-915/stats_old/: This might be interesting...
+ OSVDB-3092: /bin/stats_old/: This might be interesting...
+ OSVDB-3092: /cgi/stats_old/: This might be interesting...
+ OSVDB-3092: /mpcgi/stats_old/: This might be interesting...
+ OSVDB-3092: /cgi-bin/stats_old/: This might be interesting...
+ OSVDB-3092: /ows-bin/stats_old/: This might be interesting...
+ OSVDB-3092: /cgi-sys/stats_old/: This might be interesting...
+ OSVDB-3092: /cgi-local/stats_old/: This might be interesting...
+ OSVDB-3092: /htbin/stats_old/: This might be interesting...
+ OSVDB-3092: /cgibin/stats_old/: This might be interesting...
+ OSVDB-3092: /cgis/stats_old/: This might be interesting...
+ OSVDB-3092: /scripts/stats_old/: This might be interesting...
+ OSVDB-3092: /cgi-win/stats_old/: This might be interesting...
+ OSVDB-3092: /fcgi-bin/stats_old/: This might be interesting...
+ OSVDB-3092: /cgi-exe/stats_old/: This might be interesting...
+ OSVDB-3092: /cgi-home/stats_old/: This might be interesting...
+ OSVDB-3092: /cgi-perl/stats_old/: This might be interesting...
+ OSVDB-3092: /scgi-bin/stats_old/: This might be interesting...
+ OSVDB-17670: /clocktower/: Site Server sample files.  This might be interesting...
+ OSVDB-17670: /market/: Site Server sample files.  This might be interesting.
+ OSVDB-17670: /mspress30/: Site Server sample files.  This might be interesting...
+ OSVDB-3092: /site/iissamples/: This might be interesting...
+ OSVDB-17670: /vc30/: Site Server sample files.  This might be interesting...
+ OSVDB-3092: /_mem_bin/: This might be interesting - User Login
+ OSVDB-3092: /custdata/: This may be COWS (CGI Online Worldweb Shopping), and may be interesting...
+ OSVDB-3092: /hostingcontroller/: This might be interesting...probably HostingController, www.hostingcontroller.com
+ OSVDB-3092: /databases/: Databases? Really??
+ OSVDB-3092: /img-sys/: Default image directory should not allow directory listing.
+ OSVDB-3092: /java-sys/: Default Java directory should not allow directory listing.
+ OSVDB-3092: /javadoc/: Documentation...?
+ OSVDB-3092: /log/: Ahh...log information...fun!
+ OSVDB-3092: /manager/: May be a web server or site manager.
+ OSVDB-3092: /manual/: Web server manual found.
+ OSVDB-3092: /exchange/: This may be interesting (Outlook exchange OWA server?)...
+ OSVDB-3093: /adv/gm001-mc/: This might be interesting... has been seen in web logs from an unknown scanner.
+ OSVDB-3093: /moregroupware/modules/webmail2/inc/: This might be interesting... has been seen in web logs from an unknown
 scanner.
+ OSVDB-3093: /protected/: This might be interesting... has been seen in web logs from an unknown scanner.
```

Рисунок 3.13 – Результат сканування Nikto

```
+ OSVDB-3093: /cgi-915/ms_proxy_auth_query/: This might be interesting... has been seen in web logs from an unknown scanne
r.
+ OSVDB-3093: /bin/ms_proxy_auth_query/: This might be interesting... has been seen in web logs from an unknown scanner.
+ OSVDB-3093: /cgi/ms_proxy_auth_query/: This might be interesting... has been seen in web logs from an unknown scanner.
+ OSVDB-3093: /mpcgi/ms_proxy_auth_query/: This might be interesting... has been seen in web logs from an unknown scanner.
+ OSVDB-3093: /cgi-bin/ms_proxy_auth_query/: This might be interesting... has been seen in web logs from an unknown scanne
r.
+ OSVDB-3093: /ows-bin/ms_proxy_auth_query/: This might be interesting... has been seen in web logs from an unknown scanne
r.
+ OSVDB-3093: /cgi-sys/ms_proxy_auth_query/: This might be interesting... has been seen in web logs from an unknown scanne
r.
+ OSVDB-3093: /cgi-local/ms_proxy_auth_query/: This might be interesting... has been seen in web logs from an unknown scan
ner.
+ OSVDB-3093: /htbin/ms_proxy_auth_query/: This might be interesting... has been seen in web logs from an unknown scanner.
+ OSVDB-3093: /cgibin/ms_proxy_auth_query/: This might be interesting... has been seen in web logs from an unknown scanner
.
+ OSVDB-3093: /cgis/ms_proxy_auth_query/: This might be interesting... has been seen in web logs from an unknown scanner.
+ OSVDB-3093: /scripts/ms_proxy_auth_query/: This might be interesting... has been seen in web logs from an unknown scanne
r.
+ OSVDB-3093: /cgi-win/ms_proxy_auth_query/: This might be interesting... has been seen in web logs from an unknown scanne
r.
+ OSVDB-3093: /fcgi-bin/ms_proxy_auth_query/: This might be interesting... has been seen in web logs from an unknown scann
er.
+ OSVDB-3093: /cgi-exe/ms_proxy_auth_query/: This might be interesting... has been seen in web logs from an unknown scanne
r.
+ OSVDB-3093: /cgi-home/ms_proxy_auth_query/: This might be interesting... has been seen in web logs from an unknown scann
er.
+ OSVDB-3093: /cgi-perl/ms_proxy_auth_query/: This might be interesting... has been seen in web logs from an unknown scann
er.
+ OSVDB-3093: /scgi-bin/ms_proxy_auth_query/: This might be interesting... has been seen in web logs from an unknown scann
er.
+ OSVDB-3093: /database/: Databases? Really??
+ OSVDB-3233: /jservdocs/: Default Apache JServ docs should be removed.
+ OSVDB-3233: /akopia/: Akopia is installed.
+ OSVDB-3233: /_private/: FrontPage directory found.
+ OSVDB-3233: /_private/_vti_cnf/: FrontPage directory found.
+ OSVDB-3233: /_vti_bin/: FrontPage directory found.
+ OSVDB-3233: /_vti_bin/_vti_cnf/: FrontPage directory found.
+ OSVDB-3233: /_vti_cnf/_vti_cnf/: FrontPage directory found.
+ OSVDB-3233: /_vti_log/_vti_cnf/: FrontPage directory found.
+ OSVDB-3233: /nethome/: Netscape Enterprise Server default doc/manual directory. Reveals server path at bottom of page.
```

Рисунок 3.14 – Результат сканування Nikto

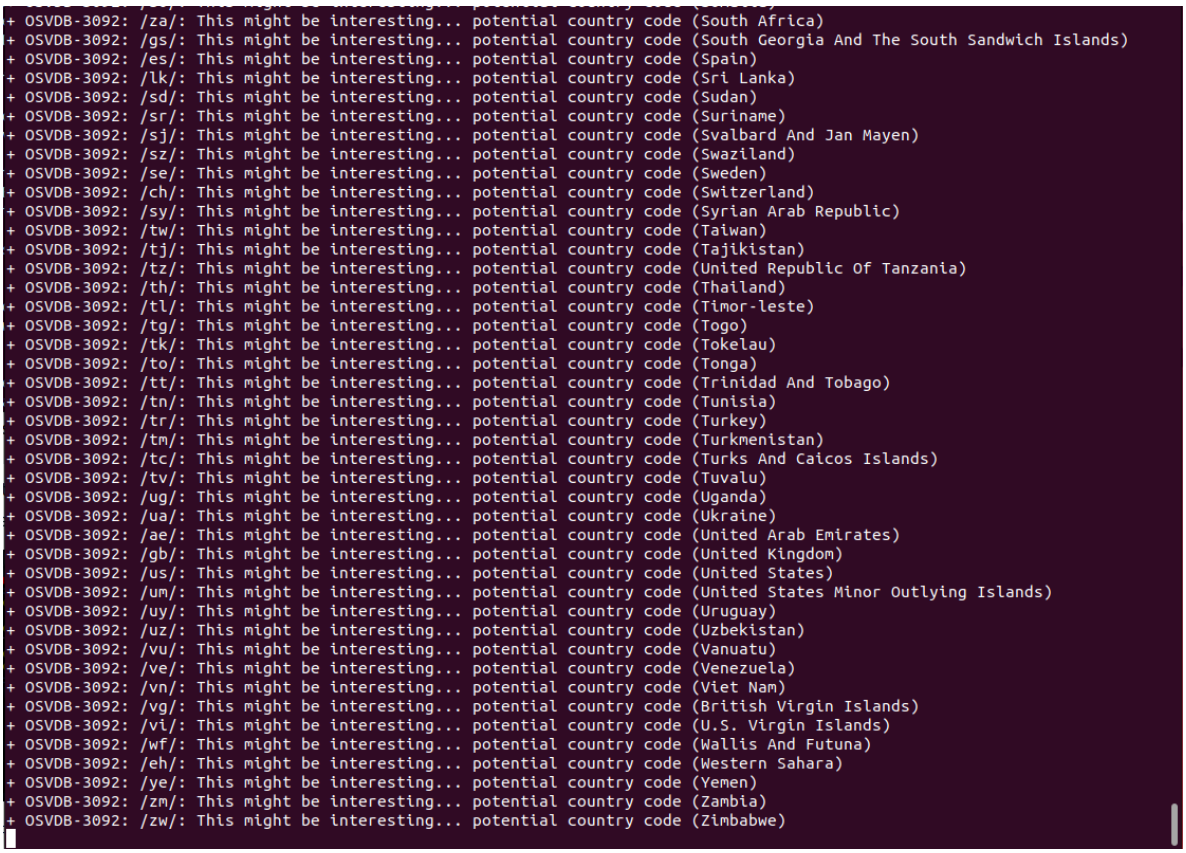


Рисунок 3.15 – Результат сканування Nikto

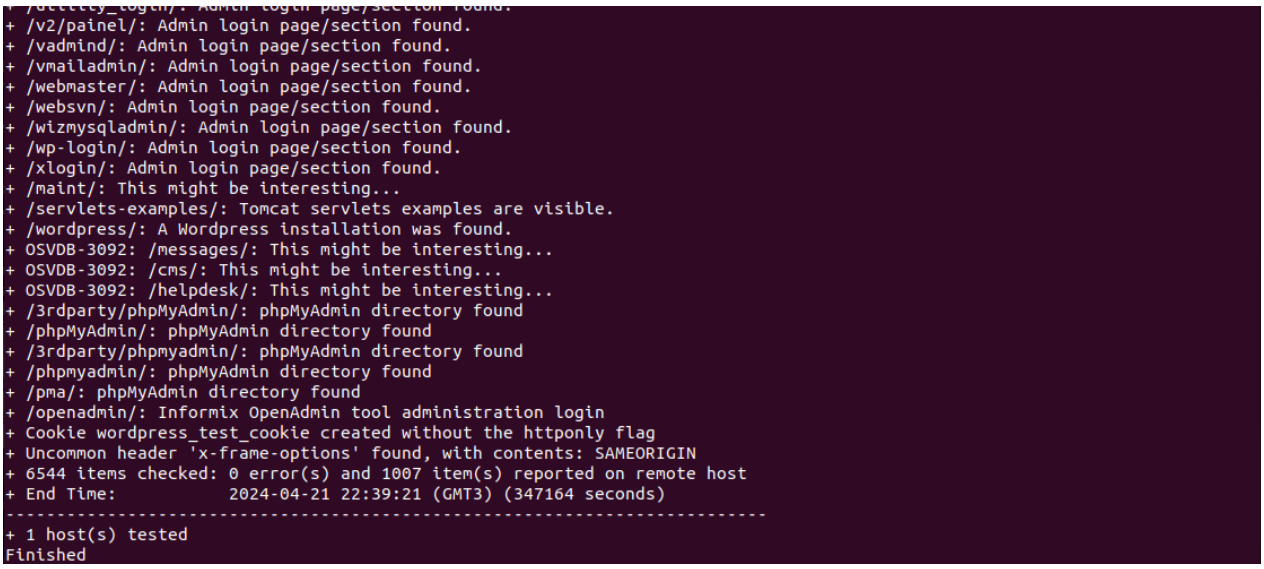


Рисунок 3.16 – Результат сканування Nikto

Наступний інструмент для виявлення можливих вразливостей - це SQLMap (рис. 3.17). Він виконує тестування вебдодатків та серверів на наявність вразливостей SQL ін’єкцій.

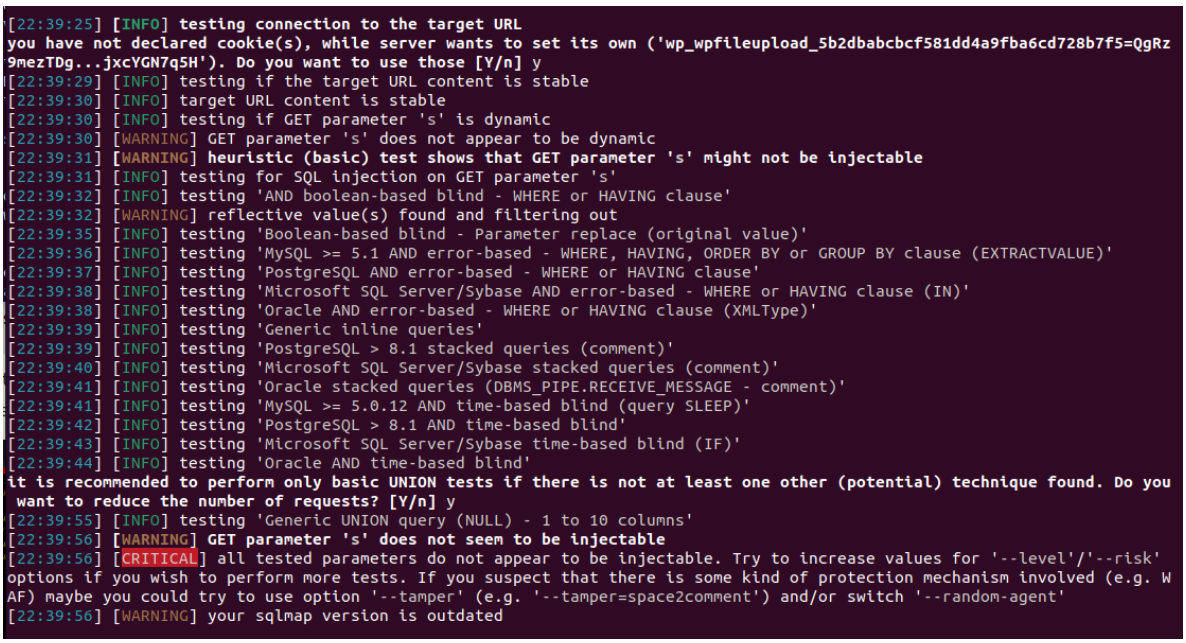


Рисунок 3.17 – Результат сканування SQLMap

Після цього, запускається утиліта WPScan. Це дуже потужне програмне забезпечення, яке надає багато можливостей у дослідженні безпеки вебдодатку і не тільки. Сканер ідентифікував 52 різні вразливості, які потенційно можуть завдати шкоди для вебдодатку на WordPress (рис. 3.18-3.24).

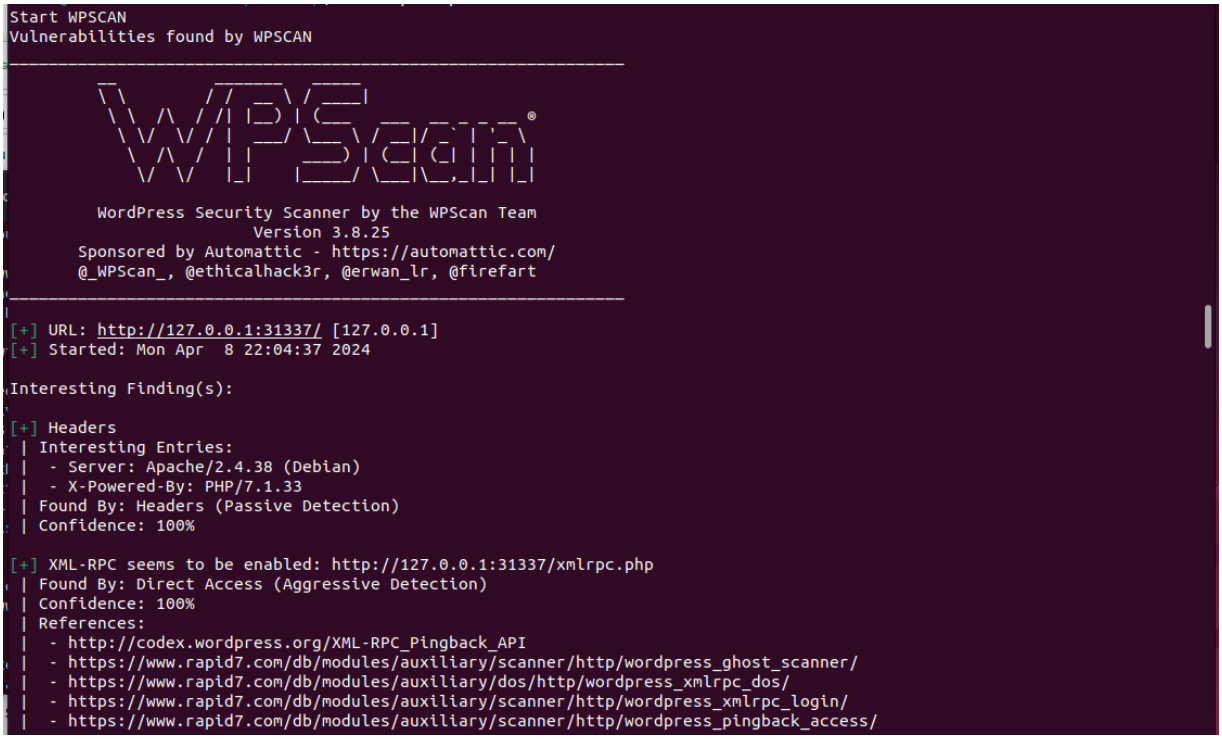


Рисунок 3.18 – Результат сканування WPScan

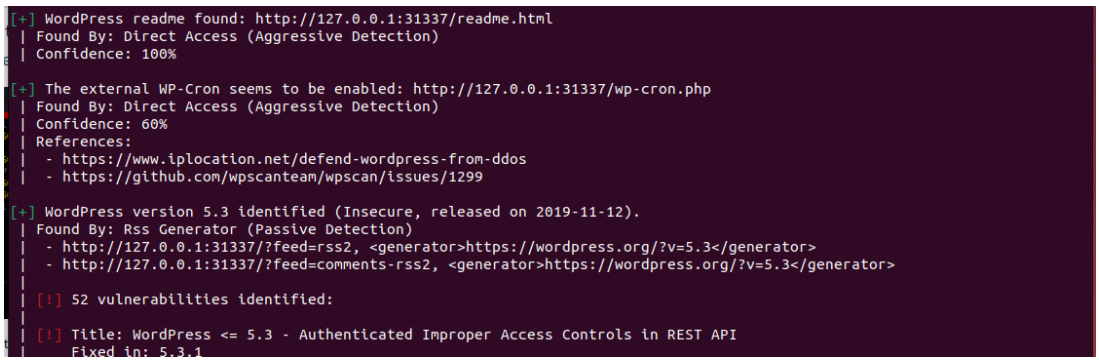


Рисунок 3.19 – Результат сканування WPScan

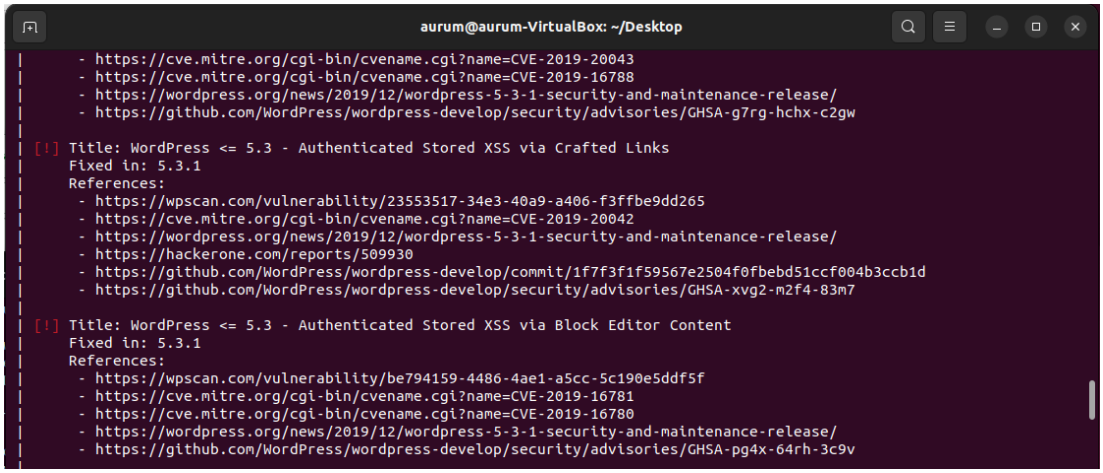


Рисунок 3.20 – Результат сканування WPScan

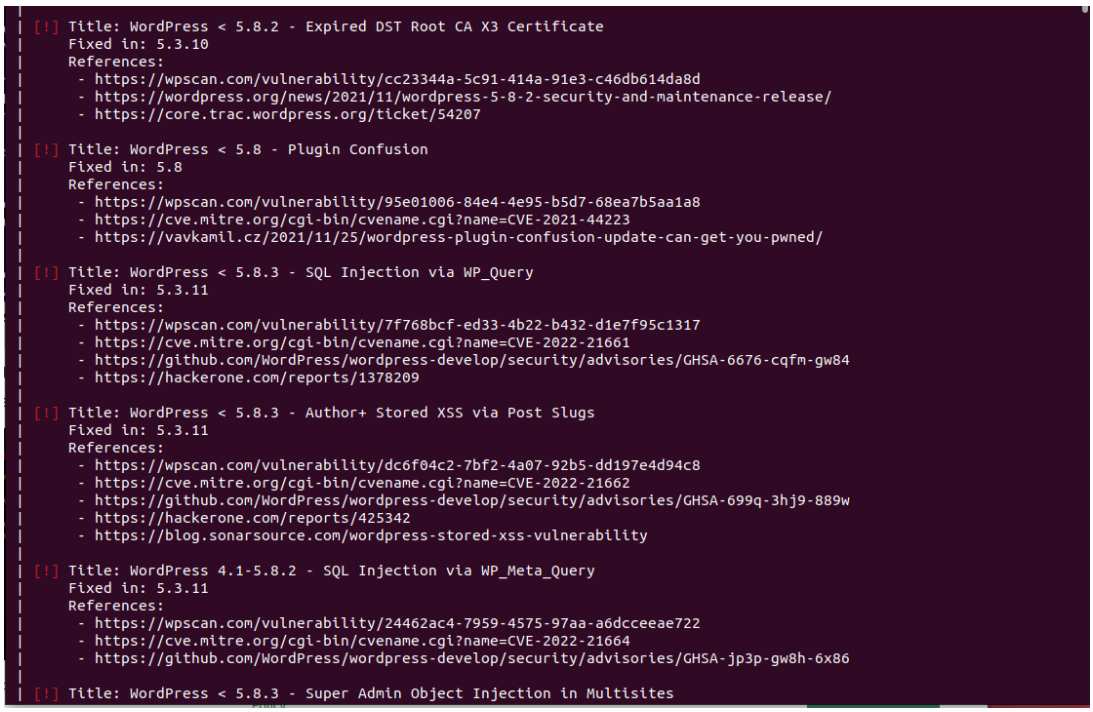


Рисунок 3.21 – Результат сканування WPScan

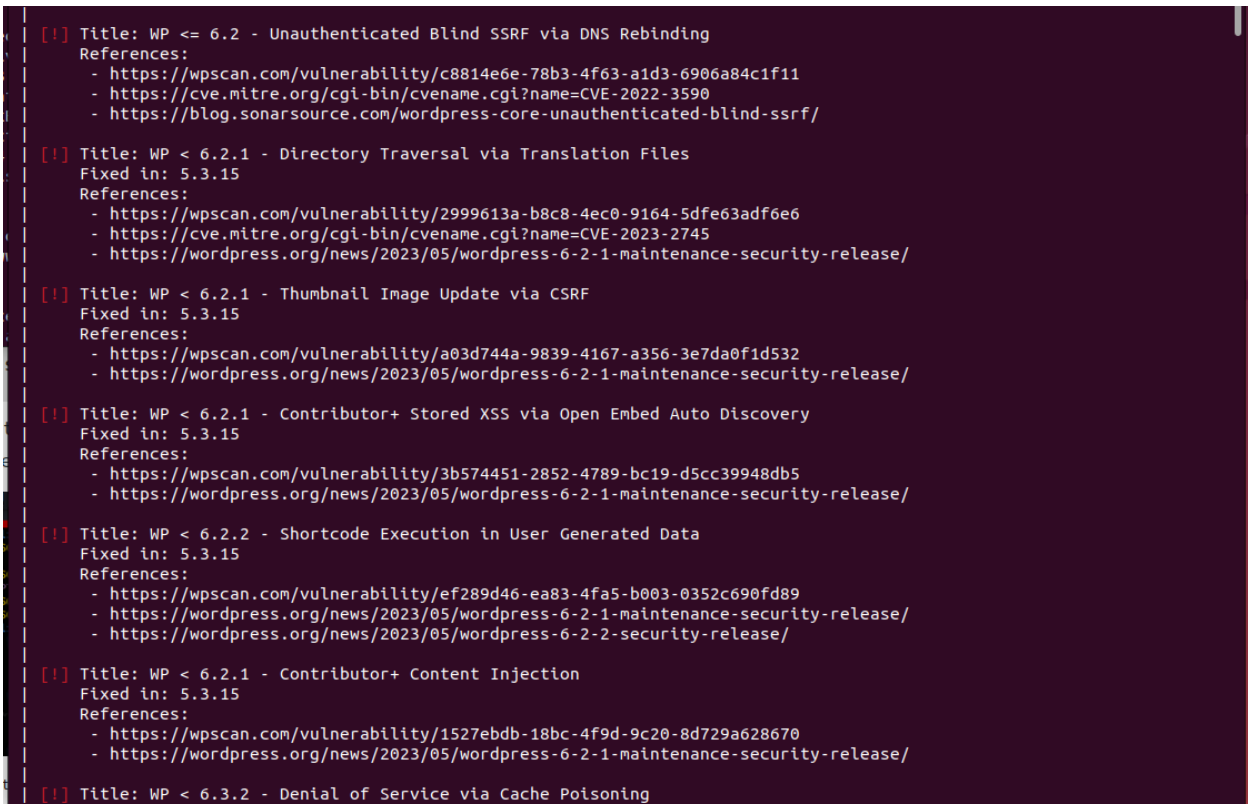


Рисунок 3.22 – Результат сканування WPScan

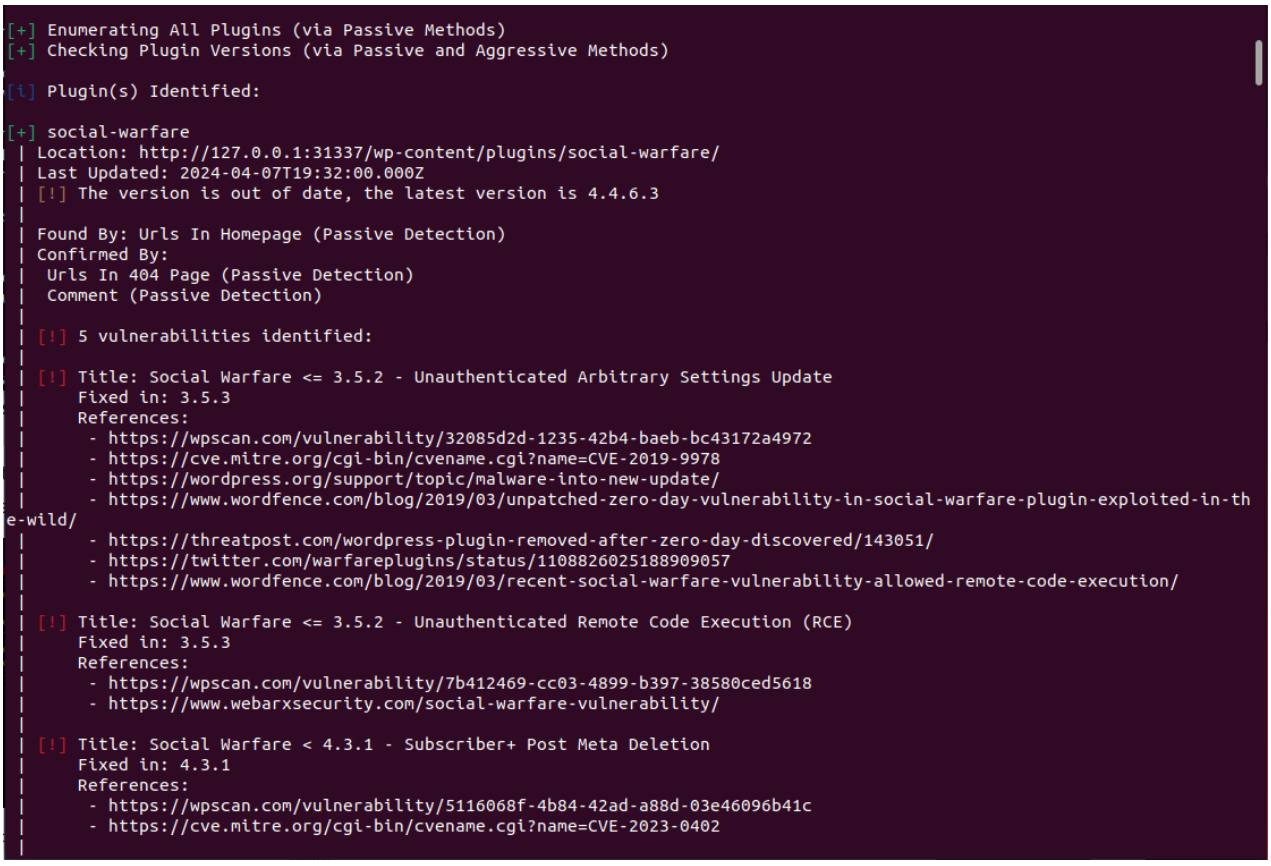


Рисунок 3.23 – Результат сканування WPScan

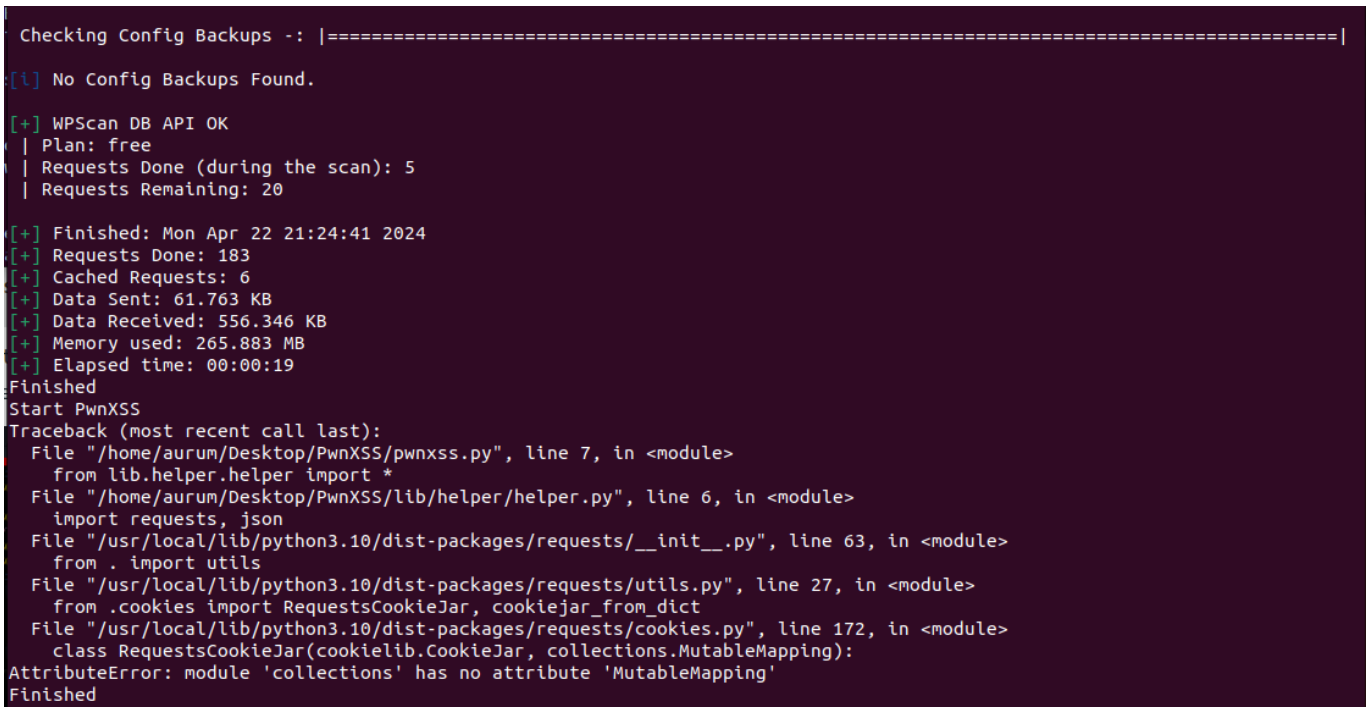


Рисунок 3.24 – Результат сканування WPScan

Останнім інструментом у дослідженні безпеки вебдодатку на WordPress - це програмне забезпечення PwnXSS, який тестує додаток на наявність можливих вразливостей міжсайтового сценарію XSS. Під час сканування він виявив потенційно цікаві посилання з різними ідентифікаторами, які можуть налічувати вразливості XSS (рис. 3.25-3.27).

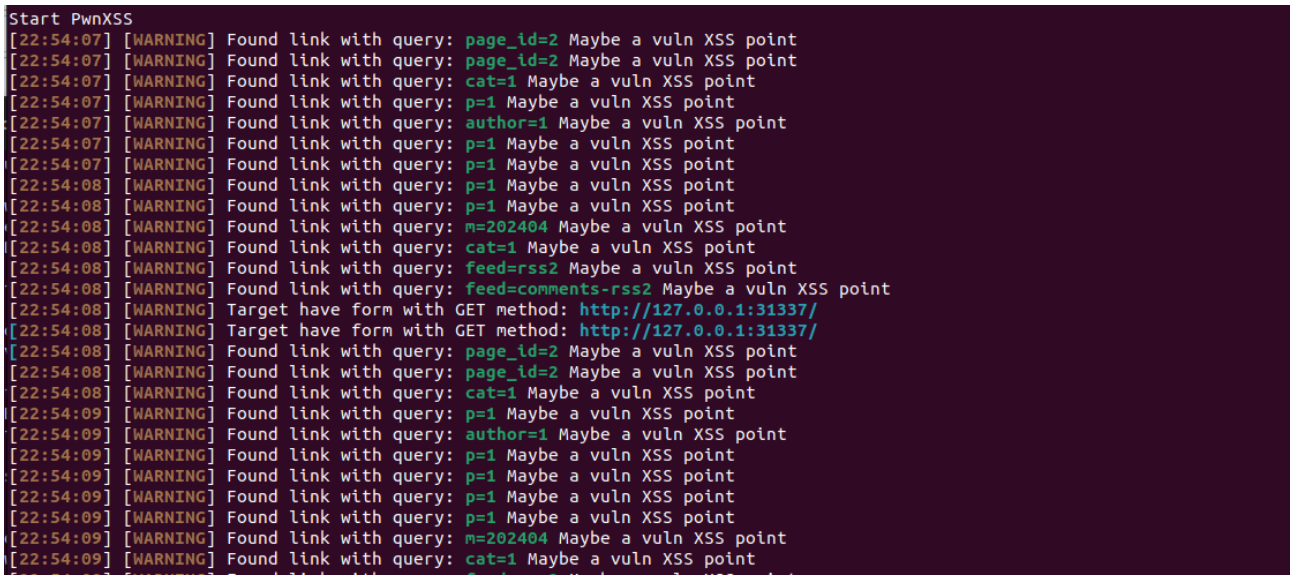


Рисунок 3.25 – Результат сканування PwnXSS

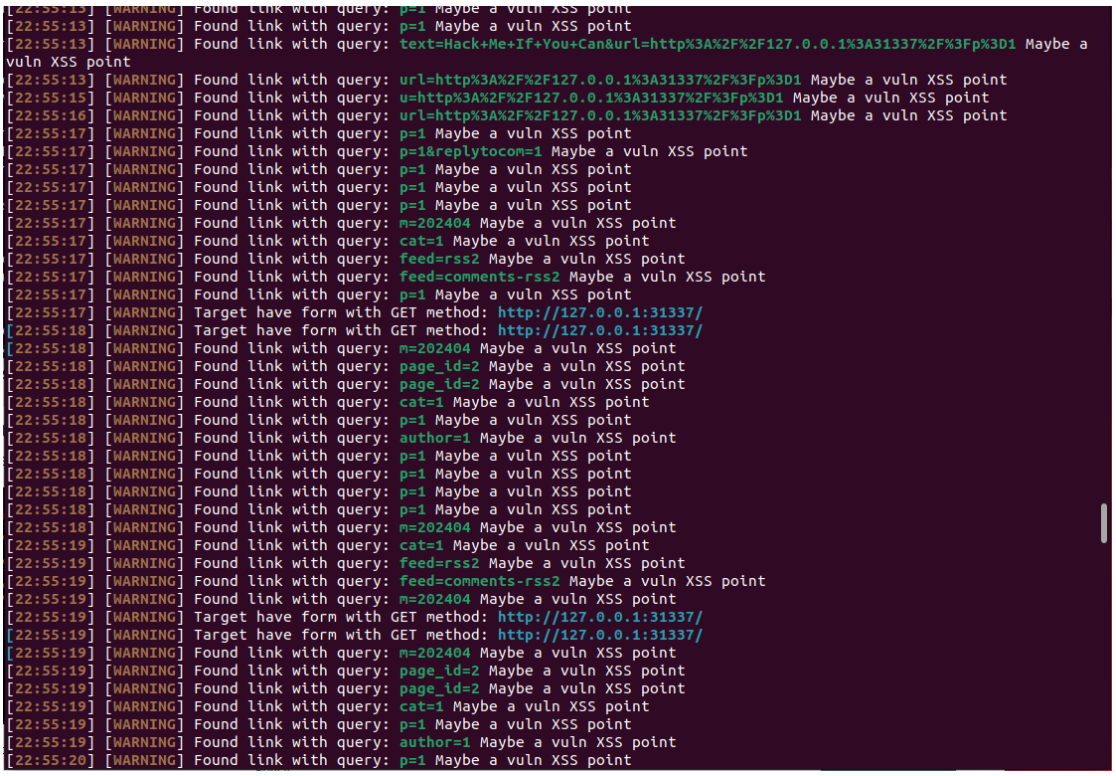


Рисунок 3.26 – Результат сканування PwnXSS

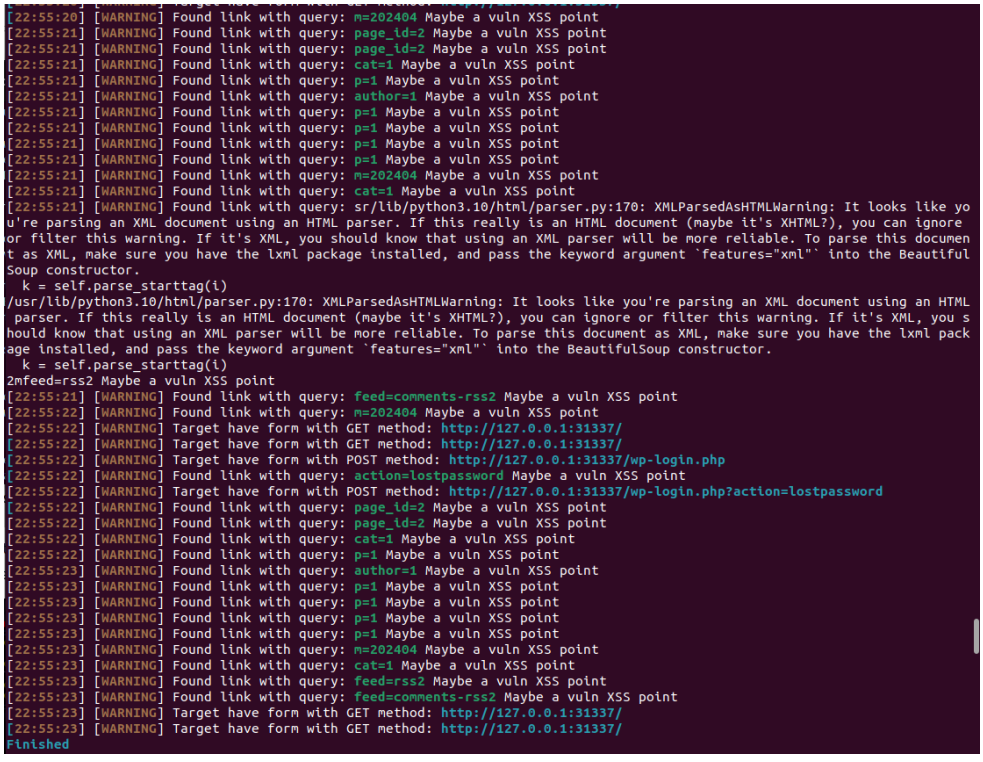


Рисунок 3.26 – Результат сканування PwnXSS

Всі результати зберігаються в окремому файлі, де зручно досліджувати та аналізувати знайдені вразливості вебдодатку (рис. 3.27).

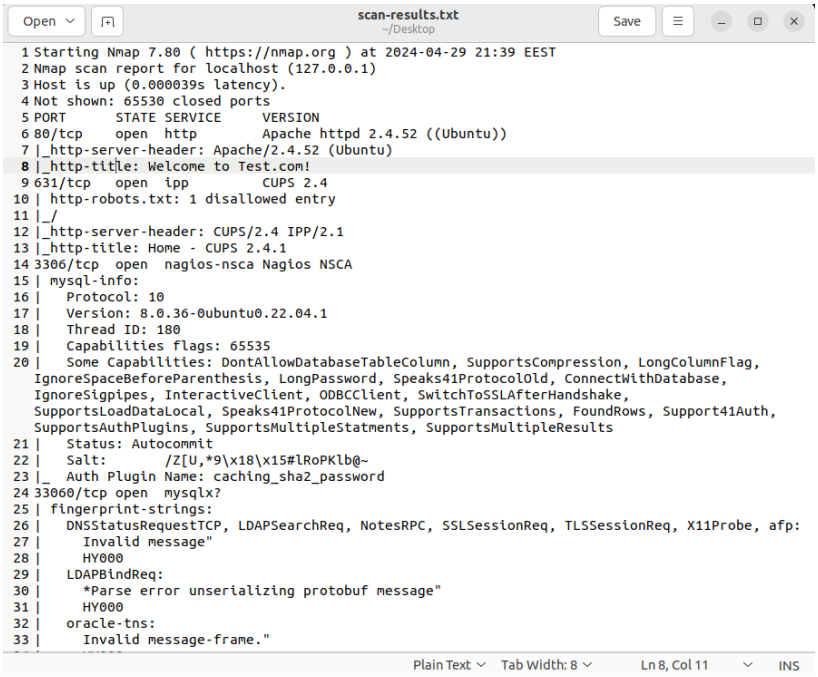


Рисунок 3.27 – Файл із результатами сканування

Висновки за розділом 3

В третьому розділі дипломної роботи було зосереджено на розробці моделі для аналізу вебексплойтів та її програмній реалізації. Модель, яку було розроблено, описує кроки, необхідні для проведення аналізу вебдодатків на наявність вразливостей. Вона включає в себе етапи збору інформації про додаток, сканування портів та вразливостей, перевірку безпеки введених даних та створення звіту.

Після розробки моделі було створено її програмну реалізацію. Було створено bin/bash скрипт, який автоматизує виконання кожного етапу моделі. Цей скрипт використовує різноманітні інструменти та методи для виявлення потенційних вразливостей у вебдодатках. Результати тестування підтвердили ефективність розробленої моделі та програмної реалізації.

Що найважливіше, цей дослідження виявило значний потенціал для підвищення рівня безпеки вебдодатків. Вдосконалення моделі та програмного забезпечення може сприяти запобіганню атакам з боку зловмисників та забезпечити більшу захищеність онлайн-платформ.

ВИСНОВКИ

Під час написання дипломної роботи було досліджено проблематику аналізу вебекплойтів на основі мови програмування JavaScript. Було виявлено, що дане питання є актуальним, так як щодня через невчасне виявлення вразливостей у вебдодатках компанії несуть чималі збитки.

Було проаналізовано методології аналізу вебдодатків на наявність вразливостей. Всі ці методології - потужні інструменти та техніки, які допомагають у різних сферах впровадження безпеки:

- OSSTMM - може бути корисною для глибокого тестування безпеки;
- Методологія OWASP надає широкий перелік рекомендацій та кращих практик з покращення безпеки вебдодатків. З огляду на широке використання платформ WordPress та OpenCart, рекомендації OWASP можуть бути дуже корисними для ідентифікації та вирішення загроз безпеці;
- Методологія WASC TC може допомогти виявити специфічні загрози, що можуть виникнути в контексті використання WordPress та OpenCart;
- Використання PTES дозволить ідентифікувати та експлуатувати вразливості, що допоможе покращити безпеку вебдодатків на платформах WordPress та OpenCart.

Окрім цього, було розроблено модель аналізу вебдодатків. Ця модель включає такі етапи: початкова розвідка вебдодатку та сканування всіх наявних сервісів, портів та служб; аналіз вебсайту на вразливості; генерація звіту.

Таким чином, подальший розвиток цієї роботи може включати розширення функціональності та оптимізацію алгоритмів для ще більш ефективного виявлення вразливостей.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. What is a Web Server and How Does it Work? [Електронний ресурс] - Режим доступу: https://www.techtarget.com/whatis/definition/Web-server

2. What is a web server? [Електронний ресурс] - Режим доступу: https://developer.mozilla.org/en-US/docs/Learn/Common_questions/Web_mechanics/What_is_a_web_server

3. Web Exploitation [Електронний ресурс] - Режим доступу: https://devopedia.org/web-exploitation

4. Експлойт - що це таке? [Електронний ресурс] - Режим доступу: https://ukeywaf.com/baza/eksplojt-shho-cze-take/

5. Що означає Експлойт? Визначення експлойту [Електронний ресурс] - Режим доступу: https://gridinsoft.ua/exploits

6. JavaScript for Hacking Made Easy: The Expert Guide on Security [Електронний ресурс] - Режим доступу: https://www.stationx.net/javascript-for-hacking/

7. The Importance Of Javascript For Web App Hacking [Електронний ресурс] - Режим доступу: https://www.cybrary.it/blog/the-importance-of-javascript-for-web-app-hacking

8. JavaScript Security [Електронний ресурс] - Режим доступу: https://jscrambler.com/learning-hub/javascript-security

9. JavaScript security [Електронний ресурс] - Режим доступу: https://snyk.io/learn/javascript-security/

10. 8 Types of Web Application Attacks and Protecting Your Organization [Електронний ресурс] - Режим доступу: https://brightsec.com/blog/8-types-of-web-application-attacks-and-protecting-your-organization/

11. The 7 Most Common Web Application Attacks And How A WAF Can Prevent Them [Електронний ресурс] - Режим доступу: https://www.nexusguard.com/blog/the-7-most-common-web-application-attacks-and-how-a-waf-can-prevent-them

12. Exploiting HTTP request smuggling vulnerabilities [Електронний ресурс] - Режим доступу: https://portswigger.net/web-security/request-smuggling/exploiting

13. Going Beyond ESLint: An Overview of Static Analysis in JavaScript [Електронний ресурс] - Режим доступу: https://www.telerik.com/blogs/going-beyond-eslint-overview-static-analysis-javascript

14. Static analysis in JavaScript: 11 tools to help you catch errors before users do [Електронний ресурс] - Режим доступу: https://blog.logrocket.com/static-analysis-in-javascript-11-tools-to-help-you-catch-errors-before-users-do/

15. Dynamic Code Analysis Tools [Електронний ресурс] - Режим доступу: https://verpex.com/blog/website-tips/dynamic-code-analysis-tools

16. Vulnerability Scanning Tools [Електронний ресурс] - Режим доступу: https://owasp.org/www-community/Vulnerability_Scanning_Tools

17. Інструменти, які підвищують якість Java коду. Мова Java та проекти з відкритим вихідним кодом [Електронний ресурс] - Режим доступу: https://javarush.com/ua/groups/posts/uk.3334.vstup-do-spotbugs-nstrument-dlja-statichnogo-analzu-kodu

18. ATTACK SIMULATION AND VULNERABILITY MANAGEMENT [Електронний ресурс] - Режим доступу: https://itbiz.ua/proizvoditeli/acunetix/acunetix-web-vulnerability-scanner-standard-premium-360/

19. Welcome to AppSpider [Електронний ресурс] - Режим доступу: https://docs.rapid7.com/appspider/

20. Top 15 JavaScript Code Analysis Tools [Електронний ресурс] - Режим доступу: https://www.appsierra.com/blog/15-javascript-code-analysis-tools

21. Як провести тестування на безпеку: посібник для Manual QA [Електронний ресурс] - Режим доступу: https://dou.ua/lenta/articles/security-testing-vulnerabilities/

22. Nmap (Man Page) [Електронний ресурс] - Режим доступу: https://nmap.org/book/man.html

23. What is Metasploit? The Beginner’s Guide [Електронний ресурс] - Режим доступу: https://www.varonis.com/blog/what-is-metasploit

24. SQL-ін’єкція [Електронний ресурс] - Режим доступу: https://hackyourmom.com/servisy/soft/sql-inyekcziya/

25. How to find SQL injection using SQLmap [Електронний ресурс] - Режим доступу: https://svyat.tech/how-to-find-sql-injection-using-sqlmap

26. Wireshark - докладний посібник з початку використання [Електронний ресурс] - Режим доступу: https://hackyourmom.com/kibervijna/wireshark-dokladnyj-posibnyk-z-pochatku-vykorystannya/

27. Ettercap [Електронний ресурс] - Режим доступу: https://www.bugcrowd.com/glossary/ettercap/

28. 5 Most Popular Web App Security Testing Methodologies [Електронний ресурс] - Режим доступу: https://www.apriorit.com/qa-blog/524-web-application-security-testing

29. Open Source Security Testing Methodology Manual [Електронний ресурс] - Режим доступу: https://www.futurelearn.com/info/courses/ethical-hacking-an-introduction/0/steps/71522

30. SECURITY BLOG OSSTMM [Електронний ресурс] - Режим доступу: https://resilientx.com/blog/osstmm-open-source-security-testing-methodology-manual-a-comprehensive-overview/

31. What Is OWASP? [Електронний ресурс] - Режим доступу: https://www.firewall.cx/security/web-application-vulnerability-scanners/what-is-owasp-introduction-to-owasp.html

32. What is OWASP? What is the OWASP Top 10? [Електронний ресурс] - Режим доступу: https://www.cloudflare.com/learning/security/threats/owasp-top-10/

33. What Is The Open Web Application Security Project (OWASP)? [Електронний ресурс] - Режим доступу: https://www.castsoftware.com/glossary/owasp

34. OWASP Dependency-Check [Електронний ресурс] - Режим доступу: https://www.hackerone.com/knowledge-center/owasp-dependency-check-how-it-works-benefits-and-proscons

35. WASC THREAT CLASSIFICATION [Електронний ресурс] - Режим доступу: https://kr-labs.com.ua/books/WASC-TC-v2_0+(1).pdf

36. Web Application Security Consortium (WASC) [Електронний ресурс] - Режим доступу: https://cybertzar.com/web-application-security-consortium-wasc

37. Penetration Testing Execution Standard (PTES) [Електронний ресурс] - Режим доступу: https://www.geeksforgeeks.org/penetration-testing-execution-standard-ptes/

38. WHAT IS THE PENETRATION TESTING EXECUTION STANDARD? [Електронний ресурс] - Режим доступу: https://blog.rsisecurity.com/what-is-the-penetration-testing-execution-standard/

39. Information System Security Assessment Framework [Електронний ресурс] - Режим доступу: https://www.futurelearn.com/info/courses/ethical-hacking-an-introduction/0/steps/71521

40. Five Methodologies That Can Improve Your Penetration Testing ROI [Електронний ресурс] - Режим доступу: https://www.eccouncil.org/cybersecurity-exchange/penetration-testing/penetration-testing-methodology-improve-pen-testing-roi/

41. PCI DSS Penetration Test Requirements [Електронний ресурс] - Режим доступу: https://pcidssguide.com/pci-penetration-test-requirements/

42. PCI PENETRATION TESTING GUIDE - ALL YOU NEED TO KNOW [Електронний ресурс] - Режим доступу: https://www.blazeinfosec.com/post/pci-penetration-testing-guide/

43. NIST SP 800-115 AND PENETRATION TESTING [Електронний ресурс] - Режим доступу: https://www.softwaresecured.com/post/nist-sp-800-115-and-penetration-testing

44. Technical Guide to Information Security Testing and Assessment [Електронний ресурс] - Режим доступу: https://nvlpubs.nist.gov/nistpubs/legacy/sp/nistspecialpublication800-115.pdf

45. MITRE ATT&CK [Електронний ресурс] - Режим доступу: https://attack.mitre.org/

46. ЯК МОДЕЛЮВАННЯ ЗЛОМІВ ТА АТАК ПОЛЕГШУЄ ВЗАЄМОДІЮ З MITRE ATT&CK [Електронний ресурс] - Режим доступу: https://iitd.com.ua/news/jak-modeljuvannja-zlomiv-ta-atak-polegshuie-vzaiemodiju-z-mitre-att-ck/

47. What Is the MITRE ATT&CK Framework? [Електронний ресурс] - Режим доступу: https://www.trellix.com/security-awareness/cybersecurity/what-is-mitre-attack-framework/

48. MITRE ATT&CK framework [Електронний ресурс] - Режим доступу: https://www.ibm.com/topics/mitre-attack

49. APT28 Group [Електронний ресурс] - Режим доступу: https://attack.mitre.org/groups/G0007/

50. APT29 Group [Електронний ресурс] - Режим доступу: https://attack.mitre.org/groups/G0016/

ДОДАТОК А

Копія наукової публікації

[1] Serhii Buchyk

Doctor of Technical Sciences, Professor at the Department of Cyber Security and Information Protection, Faculty of Information Technologies

[2] Kuroiedov Andrii

Student at the Department of Cyber Security and Information Protection, Faculty of Information Technologies

[1] Taras Shevchenko National University of Kyiv

METHODS FOR ANALYZING JAVASCRIPT-BASED WEB EXPLOITS

Abstract. Web exploits based on JavaScript are an actual problem today. JavaScript is used by a wide variety of web applications, and this popularity makes it an attractive target for attackers. This report describes protection techniques, including static and dynamic analysis, as well as recommendations for using vulnerability discovery tools.

Keywords. JavaScript, web, exploits, vulnerabilities, web sites security.

Web exploits are one of the most common security threats on the Internet today. They are used by attackers to gain unauthorized access to web applications, infect malware, steal confidential information, and even take control of entire systems.

As one of the most widely used programming languages for web development, JavaScript is part of the attack model, so it runs concurrently in the client's browser. Attackers actively exploit JavaScript vulnerabilities to create malicious code that can exploit flaws in applications and cause harm to users and organizations.

Today, such attacks as: CSRF attacks, SQL injections, XSS attacks, phishing, client-side attacks, etc.

A JavaScript-based protection model can use techniques such as code analysis for vulnerabilities, input control, and the use of various filtering mechanisms to prevent client-side attacks. The development of such a model will ensure greater security of web applications and protect users from existing threats.

CSRF attack (Cross-Site Request Forgery) - a type of attack that allows an attacker to execute a request on behalf of a user. The attack is created using a fraudulent site or a script that forces the user to perform an unwanted action on a trusted site where they are authorized [1]. Usually, the user should follow the fraudulent link that was sent to the mail. To protect against this attack, it is customary to use a token. A token is a random set of bytes that the server sends to the client and the client returns to the server. Protection is reduced to checking the token that the server generated and the token that the user sent.

SQL injections are a type of database-oriented attack in which an attacker successfully creates and inserts malicious code into a form field or url field. In turn, these lines are broadcasted to the database management system (DBMS) server for parsing and execution. If this attack is successful, an attacker can easily overcome the system's security program, access, copy, delete, and modify confidential information contained in the database [2]. The only reliable way to prevent SQL Injection is to validate input and parameterized queries, including prepared statements [3].

XSS (Cross-Site Scripting) is an attack on web applications in which an attacker inserts malicious script code or HTML code into a web page visited by a user [4]. This attack is used to inject malicious code into web pages served by other users browsers. If other users visit this page, their browsers contain embedded malicious code. To protect against XSS, programmers need to additionally use a filter and check the data that the user enters. Also need to use the Content Security Policy and check the HTTP-only and Secure flags.

The main methods for analyzing JavaScript-based attacks are static and dynamic analysis. Static analysis performs manual analysis of source code to identify vulnerabilities or optimize application performance. And dynamic analysis, in its application, uses special software for analyzing the source code, which greatly simplifies the analysis process.

In addition, during dynamic analysis can use the method of penetration testing, use additional web application scanners and use network traffic scanners.

For an example of dynamic analysis, can use the open Grabber scanner [5]. This software allows to save such vulnerabilities as XSS, SQL injection, and also analyzes the JavaScript code in the presence of existing dangerous code.

Thus, web attacks based on the JavaScript programming language are very serious threats to users and owners of web applications. Considering all the technical information, vulnerability analysis is a greater measure of countermeasures. Each of the attack protection methods is a good opportunity to protect confidential data, reputation, finances, etc., but in combination, using the most effective protection methods will help to properly protect web applications.

References:

1. ПІДРОБКА МІЖСАЙТОВИХ ЗАПИТІВ (CSRF) З ДОПОМОГОЮ FLASH. URL: https://cqr.company/ua/web-vulnerabilities/cross-site-request-forgery-csrf-via-flash/

2. What is SQL Injection (SQLi) and How to Prevent It. URL: https://www.acunetix.com/websitesecurity/sql-injection/

3. Найвідоміші вразливості веб застосунків. XSS та SQL ін’єкції, вразливості автентифікації URL: https://dou.ua/forums/topic/40613/

4. Fawaz Mahiuob Mohammed Mokbal, Wang Dan, Azhar Imran, Lin Jiuchuan, Faheem Akhtar, Wang Xiaoxi (2019). MLPXSS: An Integrated XSS-Based Attack Detection Scheme in Web Applications Using Multilayer Perceptron Technique, IEEE Access (Volume: 7), DOI: https://doi.org/10.1109/ACCESS.2019.2927417.

5. 14 best open-source web application vulnerability scanners. URL: https://resources.infosecinstitute.com/topics/application-security/14-popular-web-application-vulnerability-scanners/

ДОДАТОК Б

Копія наукової публікації

VII Міжнародна науково-практична конференція

“Проблеми кібербезпеки інформаційно-телекомунікаційних систем” (PCSITS)”

Методології аналізу вебдодатків на вразливість

Сергій Бучик[1], Куроєдов Андрій[2]

1. Кафедра кібербезпеки та захисту інформації, Київський національний університет імені Тараса Шевченка, УКРАЇНА, м.Київ, вул.Володимирська, 60, E-mail: buchyk@knu.ua

2. Кафедра кібербезпеки та захисту інформації, Київський національний університет імені Тараса Шевченка, УКРАЇНА, м.Київ, вул.Володимирська, 60, E-mail: askuroyedov@gmail.com

Web applications today face increasing cyber threats and the need for effective security measures. Researching vulnerability analysis methodologies becomes an integral part of this process. This study examines key methodologies such as OSSTMM, OWASP, WASC-TC, PTES, ISSAF, PCI, and NIST 800-115. Each of these methodologies offers its unique approach to identifying and addressing vulnerabilities, contributing to enhancing the security of web applications. Our analysis indicates that combining these methods provides a comprehensive view of web application security and ensures effective detection and mitigation of threats, ensuring the reliability and resilience of network systems.

Ключові слова - методологія, вебексплойт, вебдодатки, аналіз вебдодатків, вразливість, несанкціонований доступ, тестування на проникнення.

Вступ

В сучасному світі вебексплойти стали однією з найбільш серйозних загроз для безпеки вебдодатків та вебінфраструктурруктури. Ця проблема виникає з того, що вебексплойти дають можливість зловмисникам використовувати вразливості

вебдодатків для виконання шкідливого коду або отримання несанкціонованого доступу до системи.

Для забезпечення безпеки вебдодатків спеціалісти користуються вже наявними світовими практиками, які здатні полегшити процес аналізу безпеки вебдодатків.

Методології аналізу вебдодатків на вразливості

Дослідження вебдодатків на вразливості сьогодні - це важливе завдання, яке дозволить вебпрограмам функціонувати належним чином. Дане тестування виконують розробники та тестувальники ПЗ, які зазвичай використовують вже наявні та перевірені часом методики. До таких методологій можна віднести:

- Open Source Security Testing Methodology Manual (OSSTMM);
- Open Web Application Security Project (OWASP);
- Web Application Security Consortium Threat Classification (WASC-TC);
- Penetration Testing Execution Standard (PTES);
- Information Systems Security Assessment Framework (ISSAF);
- PCI Penetration Testing Guide;
- NIST Special Publication 800-115.

The Open Source Security Testing Methodology Manual, також відомий як OSSTMM, є методологією для проведення тестування безпеки [1].

OSSTMM виступає як орієнтир для професіоналів у галузі безпеки, етичних хакерів, аудиторів та організацій з метою створення безпечного середовища. Ця методологія широко використовується у проведенні тестування на проникнення, IT-аудиті та оцінці ризиків для підтвердження ефективності заходів безпеки [1].

Open Web Application Security Project (OWASP) - некомерційна організація, яка працює над підвищенням рівня безпеки програмного забезпечення [2].

Основна мета OWASP полягає в тому, щоб звернути увагу на найбільші загрози у сфері безпеки, з якими ми стикаємося сьогодні [2].

Організація WASC (The Web Application Security Consortium) раніше відома своєю активною роботою у сфері стандартизації безпеки вебдодатків. Проте, наразі організація не проявляє активності [3].

Класифікація загроз WASC є результатом спільної роботи з уточнення та організації загроз безпеці вебсайтів. Члени Консорціуму безпеки вебдодатків створили цей проект з метою розробки та просування галузевих стандартів термінології для опису цих проблем [3].

Penetration Testing Execution Standard (PTES) є методом тестування на проникнення, розробленим практиками інформаційної безпеки, з метою задоволення потреб у повній і актуальній стандартізації у сфері тестування на проникнення. Крім того, для ефективного управління виправленням вразливостей, він спрямований на передачу бізнесам очікувань від тестування на проникнення і керівництво ними шляхом адаптації та використання успішних проектів [4].

Структура оцінки безпеки інформаційних систем (ISSAF) ¬- це спеціалізований підхід до тестування на проникнення (Група безпеки відкритих інформаційних систем, 2006). Його обширний довідник, який налічує понад 1200 сторінок, розкриває основи цієї методології тестування. Простий підхід ISSAF легко адаптується для потреб конкретних організацій та індивідуальних тестувальників, дозволяючи створювати персоналізовані плани тестування. Будь-який тестувальник на проникнення, що використовує різні інструменти, повинен дотримуватися методології ISSAF [5].

PCI DSS Penetration Testing є типом етичного хакінгу, який симулює мережу та її цільові системи. Тестування на проникнення виходить за межі запуску автоматизованого сканера вразливостей; фахівці з безпеки проводять тести та глибоко проникають у систему [6].

Проведення тестування на проникнення PCI DSS в мережах, публічних пристроях, програмах, базах даних та інших структурах, що зберігають, обробляють або розповсюджують дані власників карток, означає, що ви намагаєтеся виявити вразливості до того, як про них дізнається зловмисник [6].

Національний інститут стандартів та технологій, або NIST, є організацією, що входить до складу Міністерства торгівлі США і має за мету бути лідером у сфері інновацій та технологій, надаючи справедливі стандарти та рішення [7].

NIST SP 800-115 - це технічний посібник з тестування та оцінки інформаційної безпеки, який використовується при плануванні та впровадженні ефективних процесів та процедур забезпечення безпеки [7].

Всі ці методології - потужні інструменти та техніки, які допомагають у різних сферах впровадження безпеки. Проаналізуємо наскільки ефективними будуть дані методики для вебдодатків, побудованих на CMS WordPress чи OpenCart.

OSSTMM зосереджується на виконанні різних типів тестів на вразливість та перевірці безпеки мережі. Для платформ WordPress та OpenCart, де безпека має велике значення, ця методологія може бути корисною для глибокого тестування безпеки.

Методологія OWASP надає широкий перелік рекомендацій та кращих практик з покращення безпеки вебдодатків. З огляду на широке використання платформ WordPress та OpenCart, рекомендації OWASP можуть бути дуже корисними для ідентифікації та вирішення загроз безпеці.

Методологія WASC TC надає систематизований підхід до ідентифікації та класифікації загроз безпеці. Вона може допомогти виявити специфічні загрози, що можуть виникнути в контексті використання WordPress та OpenCart.

PTES надає стандарт для виконання тестування на проникнення, включаючи планування, виконання тестів та документування результатів. Враховуючи широку функціональність та можливості цих платформ, PTES може бути ідеальним інструментом для систематичного тестування на проникнення. Використання PTES дозволить ідентифікувати та експлуатувати вразливості, що допоможе покращити безпеку вебдодатків на платформах WordPress та OpenCart.

Методології ISSAF та NIST Special Publication 800-115 можуть бути застосовані у реальних сценаріях для забезпечення безпеки вебдодатків на платформах WordPress та OpenCart.

Наприклад, розглянемо сценарій, де власник онлайн-магазину на платформі OpenCart прагне підвищити рівень безпеки свого вебдодатку. Використовуючи ISSAF, він може провести аналіз потенційних загроз і ризиків, що можуть виникнути внаслідок недоліків у безпеці мережі, конфігурації сервера або програмного

забезпечення. За допомогою цього аналізу він може ідентифікувати критичні вразливості і розробити стратегії для їх усунення.

Далі, використовуючи NIST Special Publication 800-115, власник магазину може впровадити стандартизовані методики тестування безпеки для проведення пенетраційних тестів та аудиту безпеки. Це дозволить виявити можливі вразливості в додатку та інфраструктурі магазину, такі як SQL-ін’єкції, XSS атаки, недоліки в конфігурації сервера тощо.

Після завершення цих процесів власник магазину може розробити індивідуалізований план безпеки, що враховує усі виявлені загрози та рекомендації з обох методологій. Цей план може включати заходи, такі як регулярне оновлення програмного забезпечення, впровадження механізмів виявлення вторгнень і вдосконалення процедур аутентифікації та авторизації.

PCI Penetration Testing Guide зосереджений на вимогах безпеки платіжних карток, але може бути корисним для загального тестування безпеки вебдодатків, які використовують модулі оплати в своєму функціоналі.

Висновок

Підводячи підсумки, можна сказати, що для покращення безпеки вебдодатків на платформі WordPress рекомендовано використовувати комбінацію різних методологій:

OWASP - для ідентифікації найбільш поширених загроз та виконання кращих практик безпеки.

OSSTMM - для глибокого тестування на вразливості та перевірки безпеки мережі.

WASC TC - для систематизації загроз та розуміння їх впливу на WordPress.

ISSAF - для комплексного аудиту та тестування на проникнення в інформаційних системах.

NIST Special Publication 800-115 - для стандартизованого підходу до пенетраційного тестування.

PCI Penetration Testing Guide - для використання методів, спрямованих на вимоги безпеки платіжних карток.

PTES - для виконання систематичного тестування на проникнення та ідентифікації вразливостей.

Комбінація цих методологій дозволить отримати повніший та комплексний погляд на безпеку вебдодатків на платформі WordPress, а також ідентифікувати та вирішувати вразливості з більшою ефективністю.

Література

[1] Open Source Security Testing Methodology Manual [Електронний ресурс] - https://www.futurelearn.com/info/courses/ethical-hacking-an-introduction/0/steps/71522

[2] What Is OWASP? [Електронний ресурс] - https://www.firewall.cx/security/web-application-vulnerability-scanners/what-is-owasp-introduction-to-owasp.html

[3] WASC THREAT CLASSIFICATION [Електронний ресурс] - https://kr-labs.com.ua/books/WASC-TC-v2_0+(1).pdf

[4] Penetration Testing Execution Standard (PTES) [Електронний ресурс] - https://www.geeksforgeeks.org/penetration-testing-execution-standard-ptes/

[5] Information System Security Assessment Framework [Електронний ресурс] - https://www.futurelearn.com/info/courses/ethical-hacking-an-introduction/0/steps/71521

[6] PCI DSS Penetration Test Requirements [Електронний ресурс] - https://pcidssguide.com/pci-penetration-test-requirements/

[7] NIST SP 800-115 AND PENETRATION TESTING [Електронний ресурс] - https://www.softwaresecured.com/post/nist-sp-800-115-and-penetration-testing

ДОДАТОК В

Код програми

Код скрипта для автоматизованого аналізу вебдодатків:

```
#!/bin/bash



echo "Start nmap"
echo "_______  ____    _    ____    _______  "
echo "|       |      / \    |    \      |    "
echo "|       |     /   \   |    /      |    "
echo "|____   |     _    |__/       |    "
echo "     |    |    /     \  |   \       |    "
echo "     |    |   /       \ |    \      |    "
echo "_____|    |  /         \|     \     |    "

nmap -p- -sV -sC -A 127.0.0.1 | tee -a "scan-results.txt"
echo "Finished"



echo "Start dirb"
echo "_______  ____    _    ____    _______  "
echo "|       |      / \    |    \      |    "
echo "|       |     /   \   |    /      |    "
echo "|____   |     _    |__/       |    "
echo "     |    |    /     \  |   \       |    "
echo "     |    |   /       \ |    \      |    "
echo "_____|    |  /         \|     \     |    "
```

```
dirb http://127.0.0.1:31337/ -f | tee -a "scan-results.txt"
echo "Finished"

echo "Start Nikto"
echo "_______  ____    _    ____    _______  "
echo "|       |      / \    |    \      |    "
echo "|       |     /   \   |     /     |    "
echo "|____       |     _    |__/       |    "
echo "      |     |    /    \  |   \      |    "
echo "      |     |   /      \ |    \     |    "
echo "_______|     |  /        \|     \    |    "

nikto -h http://127.0.0.1:31337/ | tee -a "scan-results.txt"
echo "Finished"



echo "Start SQLmap"
echo "_______  ____    _    ____    _______  "
echo "|       |      / \    |    \      |    "
echo "|       |     /   \   |     /     |    "
echo "|____       |     _    |__/       |    "
echo "      |     |    /    \  |   \      |    "
echo "      |     |   /      \ |    \     |    "
echo "_______|     |  /        \|     \    |    "

sqlmap -u 127.0.0.1:31337/?s=gngh | tee -a "scan-results.txt"
echo "Finished"



echo "Start WPSCAN"
```

```
        echo "_______  _____   _    _____   _______  "
        echo "|       |      / \    |    \     |    "
        echo "|       |     /   \   |     /    |    "
        echo "|_____      |     _    |__/       |    "
        echo "      |     |    /     \  |   \      |    "
        echo "      |     |   /       \ |    \     |    "
        echo "_______|     |  /         \|     \    |    "

        wpscan --url http://127.0.0.1:31337/ --api-token IAph8RXF1kSeypxDAsXDVrmPr3AaFsHXimatV0YeeS4 | tee -a "scan-results.txt"
        echo "Finished"

        echo "Start PwnXSS"
        echo "_______  _____   _    _____   _______  "
        echo "|       |      / \    |    \     |    "
        echo "|       |     /   \   |     /    |    "
        echo "|_____      |     _    |__/       |    "
        echo "      |     |    /     \  |   \      |    "
        echo "      |     |   /       \ |    \     |    "
        echo "_______|     |  /         \|     \    |    "

        python3 PwnXSS/pwnxss.py -u http://127.0.0.1:31337/ | grep -i "WARNING" | tee -a "scan-results.txt"
        echo "Finished"



        fi

exit 0
```