Міністерство освіти і науки України
Київський національний університет імені Тараса Шевченка

Факультет інформаційних технологій
Кафедра кібербезпеки та захисту інформації

ДОПУСТИТИ ДО ЗАХИСТУ:
завідуюча кафедри кібербезпеки
та захисту інформації
_____ Наталія ЛУКОВА-ЧУЙКО
«14» червня 2022р.

ПОЯСНЮВАЛЬНА ЗАПИСКА
дипломної роботи
бакалавра
(назва освітнього ступеня)

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

спеціальність 125 Кібербезпека
(код і назва спеціальності)

освітня програма Кібербезпека
(назва освітньої програми)

на тему: «Виявлення та ідентифікація вразливостей веб-додатків»

Виконавець: студент IV курсу, групи КБ-41
_____ **Павло ЛОВИГІН**
(підпис) (прізвище ім'я по-батькові)

	Прізвище, ініціали	**Підпис**
Керівник	Лариса МИРУТЕНКО	

Нормоконтроль	Олександр ТОРОШАНКО	

Київ 2022

Міністерство освіти і науки України
Київський національний університет імені Тараса Шевченка

Факультет інформаційних технологій
Кафедра кібербезпеки та захисту інформації

ЗАТВЕРДЖЕНО:
завідуюча кафедри кібербезпеки
та захисту інформації
_____ Наталія ЛУКОВА-ЧУЙКО
«01» листопада 2021 р.

ЗАВДАННЯ
на виконання дипломної роботи

спеціальності 125 Кібербезпека
(код і назва спеціальності)

освітньої програми Кібербезпека
(назва освітньої програми)

Студентові **КБ-41** **Павлу Олександровичу Ловигіну**
(група) (прізвище ім'я по-батькові)

Тема дипломної роботи Виявлення та ідентифікація вразливостей веб-додатків

1. ПІДСТАВИ ДЛЯ ПРОВЕДЕННЯ РОБОТИ

Тема дипломної роботи затверджена на засіданні кафедри кібербезпеки та захисту інформації протокол №5 від 29.10.2021 р.

2. ВИХІДНІ ДАНІ ДЛЯ ПРОВЕДЕННЯ РОБІТ

Основні вразливості веб-додатків, методи виявлення та ідентифікації вразливостей

3. ЗМІСТ РОЗРАХУНКОВО-ПОЯСНЮВАЛЬНОЇ ЗАПИСКИ

Необхідно ознайомитися зі структурою веб-додатків, основними вразливостями веб-додатків, порівняти методи виявлення та ідентифікації вразливостей веб-додатків, обрати методи виявлення та ідентифікації веб-додатків, розробити додаток-сканер вразливості.

4. ВИМОГИ ДО РЕЗУЛЬТАТІВ ВИКОНАННЯ РОБОТИ

Практична цінність Розроблений додаток можна використовувати для виявлення

вразливості CVE-2020-1938 і підвищення загального рівня безпеки додатку Apache Tomcat та систем, в яких використовується даний додаток.

5. ДАТА ВИДАЧІ ЗАВДАННЯ

Дата видачі завдання: 29 жовтня 2021 року

Завдання видав _____ (підпис) Лариса МИРУТЕНКО (ім'я, прізвище)

Завдання прийняв до виконання _____ (підпис) Павло ЛОВИГІН (ім'я, прізвище)

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Найменування етапів робіт	Строки виконання робіт (початок-кінець)	Відмітка про виконання
1	Уточнення постановки завдання	29.10.2021 – 26.01.2022	*виконано*
2	Аналіз літератури	27.01.2022 – 20.02.2022	*виконано*
3	Розгляд структури веб-додатків	21.03.2022 – 03.03.2022	*виконано*
4	Дослідження вразливостей	04.03.2022 – 30.03.2022	*виконано*
5	Розгляд методів виявлення та ідентифікації вразливостей	31.03.2022 – 21.04.2022	*виконано*
6	Дослідження вразливості CVE-2020-1938	22.04.2022 – 07.05.2022	*виконано*
7	Написання практичної реалізації сканера	08.05.2022 – 14.05.2022	*виконано*
8	Оформлення пояснювальної записки	15.05.2022 – 06.06.2022	*виконано*
9	Підготовка до захисту дипломної роботи	07.06.2022 – 13.06.2022	*виконано*

Завдання видав _____ (підпис) Лариса МИРУТЕНКО (ім'я, прізвище)

Завдання прийняв до виконання _____ (підпис) Павло ЛОВИГІН (ім'я, прізвище)

Термін подання дипломної роботи до ЕК 06 червня 2022 року

УДК 004.056.5

РЕФЕРАТ

Пояснювальна записка дипломної роботи складається зі вступу, трьох розділів, загальних висновків, списку використаних джерел та додатків. Основний текст займає 66 сторінки, включає в себе зміст, вступ, три розділи дипломної роботи, висновки, список джерел та 1 додаток із загальною кількістю сторінок 6. У пояснювальній записці дипломної роботи міститься 31 рисунок і 1 таблиця.

Метою роботи є виявлення та ідентифікація вразливостей веб-додатків.

Для досягнення зазначеної мети поставлено наступні завдання:

- дослідити структуру веб-додатків;
- провести аналіз найбільш поширених вразливостей;
- проаналізувати існуючі методи виявлення та ідентифікації вразливостей;
- розробити рішення для пошуку та ідентифікації вразливості типу LFI.

Об’єктом дослідження є процес виявлення та ідентифікації вразливостей, що властиві сучасним веб-додаткам.

Предметом дослідження є методи виявлення та ідентифікації вразливостей веб-додатків.

Методи дослідження: аналіз відкритих джерел, порівняння методів виявлення вразливостей, структурний аналіз, проектування додатку.

Актуальність роботи полягає в тому, що безпека саме веб-додатків є одним з найбільш важливих напрямків захисту додатків в цілому; вчасне виявлення вразливостей дозволяє значно зменшити ризики.

Практичною цінністю отриманих результатів є програмна реалізація засобу виявлення та ідентифікації вразливостей. Розроблений додаток можна використовувати для виявлення вразливості CVE-2020-1938 і підвищення загального рівня безпеки додатку Apache Tomcat та систем, в яких цей додаток використовується. У подальшому розроблений додаток можна покращити шляхом додавання нового функціоналу.

Ключові слова: веб-додаток, база даних, вразливості, виявлення, ідентифікація, сканер.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ ТА СКОРОЧЕНЬ

SAST	–	Static Application Security Testing
DAST	–	Dynamic Appication Security Testing
HTTP	–	HyperText Transfer Protocol
ПЗ	–	Програмне Забезпечення
СУІБ	–	Система Управління Інфомармаційною Безпекою
OWASP	–	Open Web Application Security Project
CWE	–	Common Weakness Enumeration
CVE	–	Common Vulnerabilities and Exposures
LFI	–	Local File Inclusion
IDOR	–	Insecure Direct Object Reference
API	–	Application Programming Interface
URL	–	Uniform Resource Locator
JWT	–	Java Web Token
CORS	–	Cross-Origin Resource Sharing
HTML	–	HyperText Markup Language
SQL	–	Standard Query Language
ORM	–	Object Relational Mapping
LDAP	–	Lightweight Directory Access Protocol
OGNL	–	Object-Graph Navigational Language
CDN	–	Content Delivery Network
SSRF	–	Server-Side Request Forgery
CSRF	–	Cross-Site Request Forgery
XSS	–	Cross-Site Scripting
XML	–	eXtended Markup Language

ЗМІСТ

ВСТУП

Веб-додатки є невід'ємною частиною життя сучасного суспільства. Наразі важко уявити себе без доступу до мережі Інтернет, в якій кожен день десятки мільйонів людей користуються веб-додатками для обміну повідомлення, пошуку інформації, замовлення товарів в Інтернет-магазинах чи для розваг. При цьому дані, які циркулюють каналами зв'язку, можуть містити особисту інформацію користувачів: номери банківських карток, паролі, номери мобільних телефонів, паспортні дані, фізичні адреси та багато іншого. Тому безпечність таких дій має бути одним з пріоритетів для розробників та власників додатків, оскільки порушення цілісності, конфіденційності або доступності інформації може призвести до несподіваних та, іноді, критично складних ситуацій.

Наприклад, у травні 2021 року було виявлено злам систем найбільшого у США нафтопровідного оператора – Colonial Pipelines [1]. Внаслідок цього у країні було введено надзвичайний стан [2], а робота компанії повністю зупинилась і була відновлена лише через тиждень, коли компанія заплатила 5 млн доларів зловмисникам. Через це ціна палива різко підскочила через дефіцит, а усе східне узбережжя США було тимчасово відрізано від подачі пального та нафтопродуктів.

Також відомі випадки витоку даних користувачів, як наприклад виток інформації з відомого українського додатку Дія [3], хоча державні органи і спростували цю інформацію [4]. В цьому інциденті зловмисники виставили на продаж особисті дані 2 мільйонів українських громадян, які користувались додатком. Серед даних були помічені паспортні дані, ім'я, адреси електронної пошти, номери мобільного телефону, адреси реєстрації та індивідуальні податкові номери.

Отже, безпека додатків, в тому числі веб-додатків, є актуальною темою на сьогоднішній час. Вчасне виявлення та виправлення вразливостей може зберегти колосальні об'єми фінансових ресурсів та дані користувачів.

РОЗДІЛ 1
СТРУКТУРА ТА ОСНОВНІ ВРАЗЛИВОСТІ ВЕБ-ДОДАТКІВ

1.1 Нормативно-правова база у сфері захисту інформації

Закон України «Про інформацію» [5] визначає терміни і поняття, пов’язані з інформацією і інформаційними відносинами. Закон визначає різницю між документом і інформацією. Так, інформація – відомості чи дані, що можуть бути збережені на матеріальних носіях, а документ – сам матеріальний носій, що зберігає інформацію.

Закон визначає основні принципи інформаційних відносин. Основними принципами є:

- гарантованість права на інформацію;
- відкритість, доступність інформації, свобода обміну інформацією;
- достовірність і повнота інформації;
- свобода вираження поглядів і переконань;
- правомірність одержання, використання, поширення, зберігання та захисту інформації;
- захищеність особи від втручання в її особисте та сімейне життя;

Суб’єктами інформаційних відноси є фізичні особи, юридичні особи, об'єднання громадян, суб'єкти владних повноважень. Об’єктом інформаційних відносин є інформація. З точки зору веб-додатку суб’єктами інформаційних відносин є користувачі веб-додатку, власники веб-додатку і державні органи, що регулюють їхні відносини. Об’єктом є інформація, що циркулює у веб-додатку.

Закон класифікує інформацію з обмеженим доступом на конфіденційну, таємну та службову. Конфіденційною є інформація про фізичну особу; інформація, доступ до якої обмежено фізичною або юридичною особою, крім державних органів. Конфіденційна інформація може поширюватися за бажанням (згодою) відповідної особи у визначеному нею порядку. Для веб-додатку конфіденційною є інформація про паспортні дані користувачів, ідентифікаційні номери платників податків, адреси

проживання тощо. До інформації з обмеженим доступом не може бути внесена інформація про стан довкілля; якість харчових продуктів і предметів побуту; аварії, катастрофи, небезпечні природні явища та інші надзвичайні ситуації, що загрожують безпеці людей; стан здоров'я населення, його життєвий рівень; факти порушення прав і свобод людини, включаючи інформацію із архівних документів; незаконні дії органів державної влади, органів місцевого самоврядування, їх посадових та службових осіб; іншу інформацію, що регулюється законами України і міжнародними домовленостями.

Закон України «Про захист персональних даних» [6] регулює правові відносини, пов’язані із захистом і обробкою персональних даних і спрямований на захист права людини на невтручання у особисте життя. Закон визначає персональні дані як відомості чи сукупність відомостей про фізичну особу, яка ідентифікована або може бути конкретно ідентифікована. Термін «суб’єкт персональних даних» означає фізичну особу, чиї особисті дані обробляються. З точки зору веб-додатку суб’єктом персональних даних є користувач системи. Термін «володілець персональних даних» означає фізичну або юридичну особу, яка визначає мету обробки персональних даних, встановлює склад цих даних та процедури їх обробки. Для веб-додатку розпорядником персональних даних є людина чи компанія, яка володіє веб-додатком, що надає послуги користувачам і збирає особисті дані за їхньої згоди.

Закон визначає права суб’єкта обробки персональних даних, особливості повідомлення про обробку персональних даних, використання персональних даних, підстави для обробки персональних даних, дії щодо збирання, накопичення та зберігання, поширення, видалення чи знищення персональних даних, порядок доступу до персональних даних, умови відстрочення або відмови у доступі до персональних даних, оскарження рішення про відстрочення або відмову у доступі до персональних даних, процес повідомлення про дії з персональними даними, забезпечення захисту персональних даних, відповідальність за порушення законодавства.

Положення «Про організацію заходів із забезпечення інформаційної безпеки в банківській системі України» [7] встановлює обов'язкові мінімальні вимоги щодо

організації заходів із забезпечення інформаційної безпеки та кіберзахисту; принципи управління інформаційною безпекою; вимоги до інформаційних систем банку, що взаємодіють з інформаційними системами Національного банку України, з урахуванням напрямів розвитку криптографічного захисту інформації в інформаційних системах Національного банку. Документ містить вимоги щодо системи управління інформаційною безпекою (СУІБ), криптографічного захисту інформації та інших заходів безпеки щодо інформаційних систем. Для веб-додатків, чия діяльність не пов’язана із інформаційними системами національного банку, документ може розглядатися як рекомендація із покращення захищеності системи.

Стандарт NIST 800-63B [8] визначає вимоги для федеральних агентств з упровадження служб цифрової ідентифікації. Для веб-додатків, чия робота не стосується діяльності федеральних агентств, стандарт може бути використаний як набір рекомендацій щодо аутентифікації суб’єктів інформаційної діяльності.

Стандарт RFC 2616: Hypertext Transfer Protocol – HTTP/1.1 [9] визначає діяльність, способи застосування протоколу HTTP версії 1.1. Документ визначає HTTP як загальний протокол прикладного рівня для обміну інформацією у форматі гіпертексту. Протокол може бути використаний не лише для гіпертекстових даних за рахунок розширення його методів, кодів помилок, заголовків.

Стандарт ISO 27000 «Інформаційні технології – Методи і засоби забезпечення безпеки – Системи управління інформаційною безпекою – Огляд і термінологія» [10] визначає основні терміни інформаційної безпеки, що будуть використовуватися у даній роботі.

1.2 Загальна структура веб-додатків

Архітектура веб-додатків визначає взаємодію між додатками, базами даних та іншими сервісами для забезпечення сумісної роботи декількох додатків як одна система.

Веб-додатки базуються на клієнт-серверній взаємодії. Клієнтом виступає прикладна програма, яка виконується на апаратному забезпеченні користувача. Клієнт надсилає запити відповідно до введених користувачем даних і повертає відповідь, яку отримує від сервера. Сервером є прикладне програмне забезпечення, яке виконується на апаратному забезпеченні власника веб-додатку (або у хмарному середовищі, яке він орендує). Сервер обробляє запити клієнта та відповідно до них повертає відповіді. Під час обробки запитів сервер може звертатись до баз даних чи файлової системи. Сучасні додатки зазвичай являють собою комплекс з декількох програм, які обмінюються між собою.

Зазвичай для написання програм на боці веб-серверу використовуються наступні мови програмування [11]:

- Ruby on Rails;
- PHP;
- C#;
- Java;
- Python;
- JavaScript.

В цілому програма може бути реалізована будь-якою мовою програмування, яка надає можливості обробки HTTP-запитів та створення HTTP-відповідей.

З боку клієнта зазвичай використовуються мови CSS, JavaScript та HTML.

Залежно від числа серверів та методів зберігання даних можна виділити такі моделі архітектури веб-додатків [12]:

- один веб-сервер та одна база даних;
- декілька веб серверів та одна база даних;
- декілька веб-серверів та декілька баз даних;
- мікросервісна модель.

Найпростішою моделлю є використання одного веб-сервера та однієї бази даних. Така модель має переваги простоти впровадження та обслуговування веб-додатку, але

є (відносно інших) менш надійною, оскільки вихід з ладу веб-сервера за тих чи інших причин призведе до порушення доступності інформації з усього веб-додатку.

Використання декількох веб-серверів і однієї бази даних спрямоване на підвищення надійності веб-додатку – якщо один з веб-серверів вийде з ладу, то інші отримають перенаправлений запит користувача і повернуть відповідь; також зменшується навантаження на веб-сервери шляхом рівномірного розподілу запитів користувачів. Найбільш потенційно проблемним місцем в такій моделі стає база даних, оскільки якщо втратиться доступність сховища даних, то веб-додаток перестане функціонувати незалежно від кількості веб-серверів.

Найбільш надійною схемою є використання декількох веб-серверів і декількох баз даних. Бази даних можуть використовуватись двома шляхами: повністю дублювати одна одну на випадок виходу з ладу однієї з них та рівномірно розподіляти інформацію між ними. Перший шлях надає більшу надійність, а другий зменшує навантаження на кожну окрему базу даних. До недоліків можна віднести складність впровадження та обслуговування такого веб-додатку.

З боку способу реалізації функціональних частин більшість додатків можна поділити на ті, що використовують монолітну архітектуру і ті, які використовують мікросервісну. Існують і інші види архітектури веб-додатків, але вони менш вживані і в чомусь є покращеними версіями мікросервісної та монолітної. До монолітної структури можна віднести перші три моделі з вищезгаданих.

При використанні монолітної архітектури різні функціональні частини (система аутентифікації та авторизації, бізнес-логіка тощо) реалізуються в одній програмі.

Така архітектура має наступні переваги:

- простота створення і обслуговування додатку;
- простота впровадження сторонніх компонентів; простота впровадження у використання.

У такої архітектури існують наступні недоліки:

- велика кількість програмного коду в одному додатку, що ускладнює роботу з ним та розуміння принципів роботи програми;
- складність оновлення додатку;
- неможливість швидкого масштабування додатку;
- ускладнення переходу на нові версії або нові технології через залежність одних елементів додатку від інших.

Мікросервісна архітектура характеризується розподіленням на відокремлені функціональні елементи, кожен з яких виконує лише свою невелику частину функцій додатку.

Існують наступні переваги:

- легкість розуміння принципів роботи програми завдяки розподіленню коду та функцій по різних сервісах;
- незалежність сервісів один від одного, що надає можливість додавати нові або модифікувати старі компоненти веб-додатку;
- помилки одного сервісу не впливають на інші;
- легкість масштабування;
- можливість використання різних технологій для кожного з сервісів.

Основними недоліками є:

- складність створення таких веб-додатків, оскільки кожен сервіс має бути розроблено незалежно від інших і поєднано в одну систему;
- підвищення вартості розробки;
- підвищені вимоги до конфігурації і обслуговування кожного окремого сервісу.

1.3 Основні вразливості веб-додатків

Вразливості в безпеці – це нездатність системи протистояти реалізації певної загрози або сукупності загроз [13]. Іншими словами, це недоліки реалізації

програмного забезпечення, завдяки яким можна порушити цілісніть системи і призвести до неправильної роботи системи.

Найпоказовішим та найбільш використовуваним документом щодо найбільш поширених вразливостей веб-додатків є OWASP Top Ten [14], що створено організацією Open Web Application Security Project. На додаток до нього існує список CWE (Common Weakness Enumeration) [15] відносно вразливостей програмного забезпечення загалом та CVE (Common Vulnerabilities and Exposures) [16], що містить конкретні вразливості конкретного ПЗ.

1.3.1 OWASP

Версія документа 2021 року виділяє наступні категорії вразливостей:

1. Порушений контроль доступу – контроль доступу забезпечує політику, за якої користувачі не можуть діяти за межами їх дозволів. Порушення зазвичай призводять до несанкціонованого доступу до інформації, модифікації або видалення даних.

Поширені вразливості контролю доступу включають у себе:

- порушення принципу найменших привілеїв або заперечення доступу за замовчуванням, коли доступ має надаватись лише для певних конкретних ролей або користувачів, але надається всім;
- обхід перевірок контролю доступу шляхом змінення URL-адреси, внутрішнього стану додатку або HTML-сторінки, змінення запитів до API;
- дозвіл на перегляд або редагування чужого облікового запису за його унікальними ідентифікатором (IDOR – insecure direct object references);
- дозвіл до API з відсутнім контролем доступу методів POST, PUT та DELETE;
- підвищення привілеїв – виконання дій від імені користувача не ввійшовши в систему або від імені адміністратора при вході як звичайний користувач;
- маніпуляції з метаданими, такими як підробка або відтворення чужого токену контролю доступу - JSON Web Token (JWS) або файлу cookie;

- неправильна конфігурація CORS (Cross-Origin Resource Sharing), що дозволяє доступ з неавторизованих або недовірених джерел походження;
- перехід на привілейовані сторінки як неаутентифікований користувач.

Існують такі рекомендації щодо запобігання вразливостей контролю доступу:

- контроль доступу є ефективним лише у довіреному програмному коді на боці сервера або безсерверного API, де атакуючий не зможе модифікувати метадані та обійти контроль доступу;
- закриття доступ до усіх сторінок, крім публічних (так званий deny by default);
- впровадження механізмів контролю доступу лише один раз і їх повторне використання у всьому додатку, включаючи мінімізацію використання CORS;
- вимикання листингу (надавання списку) директорій веб-серверу та вдостовіритись, що метадані та файли для резервного відновлення не присутні в коренних директоріях серверу;
- повідомлення адміністраторів у випадку непроходження контролю доступу, коли це доречно (наприклад, повторні випадки);
- обмеження швидкості роботи API;
- ідентифікатори сесії мають становитись недійсними одразу після виходу користувача з системи, де це можливо. Якщо необхідно використання більш тривалих ідентифікаторів, то слід використовувати стандарт OAuth для відкликання доступу.

2. Криптографічні порушення – використання слабких або застарілих протоколів шифрування або невикористання криптографічних методів захисту інформації взагалі. Найбільш поширеними причинами для такого типу вразливостей є використання застарілого TLS/SSL чи власноруч створені реалізації криптографічних механізмів, які використовують слабкі ключі або повторне їх використання. Для запобігання слід мінімізувати збереження чутливої інформації на сервері та впевнитись у використанні лише стандартних сучасних алгоритмів та протоколів шифрування.

3. Ін’єкції – виникають коли дані, що надаються користувачем неправильно або взагалі ніяк не фільтруються додатком. Серед найбільш поширених ін’єкцій можна

виділити SQL, NoSQL, OS command, ORM, LDAP, OGNL. Наслідком атаки може стати неавторизований доступ до інформації, модифікація чи видалення інформації, можливість виконання власного коду на сервері атакуючим. Для запобігання рекомендується використовувати лише безпечні API, які не використовують інтерпретатори; валідувати та фільтрувати усі дані, які надаються користувачем; використовувати вбудовані обмеження (для SQL це LIMIT) для запитів, які виконуються у додатку.

4. Небезпечний дизайн – широка категорія, яка включає в себе різні вразливості, які пов’язані з відсутнім або неефективним дизайном контролю. Зазвичай в цю категорію відносять вразливості, що пов’язані з використанням небезпечних конструкцій у коді додатку. Наприклад, використання вразливих функцій. Для запобігання рекомендується використовувати методології безпечного програмування додатків та Secure Development Lifecycle (SDLC), здійснювати постійні періодичні перевірки безпечності створених додатків та коду.

5. Недоліки конфігурації безпеки – додаток може бути вразливим якщо:

- увімкнені або встановлені непотрібні та не використовувані функції (відкриті непотрібні порти, служби, сторінки, облікові записи або привілеї);
- облікові записи за замовченням та їх паролі за не змінені та працюють;
- обробка помилок розкриває набагато більше інформації, ніж того потребують користувачі (наприклад, дамп стеку коли достатньо повідомлення «Сталася помилка.»);
- функції безпеки не використовуються або налаштовані неправильно;
- налаштування безпеки у серверах чи фреймворках (наприклад Apache Struts, Spring) не встановлені на безпечні значення;
- сервер не використовує заголовки, які пов’язано з безпекою;

Для запобігання рекомендується не встановлювати та не використовувати зайві не потрібні функції, запровадити процес загартовування безпеки системи, переглянути та оновити конфігурації відповідно до всіх оновлень та нотаток щодо безпеки.

6. Використання вразливих та застарілих компонентів – використання таких компонентів може призвести до багатьох небажаних ефектів. Для запобігання рекомендуються постійно оновлювати всі компоненти системи, коли з’являються нові версії ПЗ, відмовитись від використання вразливих компонентів, якщо для них нема оновлень та постійно слідкувати за цими процесами.

7. Порушення ідентифікації та аутентифікації - вразливості, експлуатація яких дозволяє зловмиснику видавати себе за легітимного користувача. Зазвичай існують наступні причини появи таких вразливостей:

- існують можливості проведення автоматизованих таких атак, як атаки грубої сили або атаки за словником;
- додаток дозволяє використання слабких паролів та паролів за замовченням;
- використовуються слабкі або неефективні процеси відновлення паролів;
- незахищені криптографічним чином паролі, як під час руху, так і під час зберігання (перекликається з категорією 2 – криптографічні порушення);
- неефективна мультифакторна аутентифікація;
- ідентифікатор сесії викривається в URL (наприклад, www.site.com/page.php?userid=1);
- повторне використання та використання небезпечних ідентифікаторів сесії.

Для запобігання рекомендується використовувати мультифакторну аутентифікацію, перевірки паролів на надійність, не використовувати паролі за замовченням, обмежувати кількість спроб входження в систему, використовувати безпечні менеджери сесії на боці серверу.

8. Порушення цілісності даних та програмного забезпечення – вразливості, пов’язані з використанням коду та інфраструктури, які не захищено від порушень цілісності. Наприклад, використання плагінів, бібліотек або модулів з недовірених джерел, репозиторіїв та мереж доставки контенту (CDN). Використання функції автооновлення, яка не вбачає достатніх перевірок цілісності оновлень також може стати причиною появи вразливості. Для запобігання рекомендується:

- використовувати цифрові підписи або схожі механізми контролю цілісності та походження ПЗ або даних;
- переконатись що бібліотеки та залежності отримуються з довірених репозиторіїв;
- запровадити процес перевірки коду та змін конфігурації для мінімізації ймовірності включення шкідливого коду або налаштування до використовуваного програмного забезпечення.

9. Порушення моніторингу та журналювання – невикористання ефективних механізмів моніторингу та журналювання унеможливлює реагування на інциденти. Для запобігання слід переконатись у наступному:

- журналюються всі спроби входження в систему та надання доступу;
- журнали подій генеруються у зручному для аналізу форматі;
- журнали подій правильно закодовані для унеможливлення ін’єкцій або атак на системи моніторингу.

10. Підробка запитів на боці серверу (SSRF) – виникають коли веб-додаток звертається до стороннього зовнішнього ресурсу без перевірки наданого користувачем URL. Це дозволяє зловмиснику надіслати дані у будь-яку потрібну йому частину системи, навіть якщо вона захищена фаєрволом чи іншим типом контролю мережевого доступу. Для запобігання рекомендується:

- виділити функціонал доступу до віддалених ресурсів у окремі мережі;
- обмежити весь трафік, що йде на вихід з системи, крім потрібного, за допомогою фаєрволів;
- валідувати всі дані, надані користувачем;
- відключити HTTP перенаправлення.

1.3.2 CWE

CWE – common weakness enumeration – список типів (видів) вразливостей програмного та апаратного забезпечення, який створено спільнотою дослідників безпеки та компанією MITRE. Список є набагато ширшим за OWASP Top Ten.

Список найбільш небезпечних вразливостей за 2022 рік складається з 25 позицій та має наступний вигляд [17]:

1. неправильне обмеження операцій у межах буфера пам’яті;
2. неправильна нейтралізація даних, наданих користувачем під час генерації веб-сторінок (XSS);
3. неправильна перевірка вводу;
4. викриття чутливої інформації;
5. читання за межами пам’яті;
6. неправильна нейтралізація спеціальних елементів, які використовуються у командах SQL (SQL-ін’єкції);
7. Use After Free – використання вказівника на ячейку пам'яті після її очищення;
8. переповнення цілого числа;
9. міжсайтова підробка запитів (CSRF – cross-site request forgery);
10. неправильне обмеження доступу до закритих директорії (Path Traversal);
11. неправильна нейтралізація спеціальних елементів, які використовуються в командах операційної системи (OS Command Injection);
12. запис за межами пам’яті;
13. неправильна аутентифікація;
14. дереференція null-вказівника;
15. неправильне призначення доступу до критичних ресурсів;
16. необмеження завантаження файлі небезпечного типу (.sh, .exe і т.д.);
17. неправильне обмеження зовнішніх сутностей XML (XML External Entity);
18. неправильний контроль генерації коду (ін'єкція коду);

19. використання «жорстко закодованих» облікових даних;

20. неконтрольоване споживання ресурсів;

21. відсутність звільнення ресурсів після виходу часу дії;

22. небезпечні пошукові шляхи;

23. десеріалізація ненадійних даних;

24. неправильне управління привілеями;

25. неправильна перевірка сертифікатів.

Список CWE є більш обширним, ніж список OWASP, оскільки включає вразливості, властиві не лише веб-додаткам. Списки мають декілька спільних елементів:

- ін’єкції (OS Command, SQL, XSS);
- порушена аутентифікація;
- викриття чутливої інформації;
- зовнішні сутності XML (XML External Entities);
- порушення контролю доступу;
- неналежні конфігурації безпеки;
- проблеми десеріалізації.

1.3.3 CVE

CVE – Common Vulnerabilities and Exposures – каталог всіх публічно розголошених вразливостей безпеки програмного та апаратного забезпечення. Станом на 05.2022 налічує 176345 записів. Метою проекту є ідентифікація, описування та внесення у каталог всіх публічно розголошених вразливостей заради підвищення загальної інформованості щодо вразливостей та безпечності використання певних версій програмного забезпечення. Кожній вразливості виділяється свій власний запис, який складається з номеру вразливості у форматі «CVE-рік-номер» (наприклад, CVE-

2020-1938 – вразливість, додана 1938-ю у 2020 році), опису вразливості, вказання версій ПЗ, які є вразливими, типом вразливості та шляхами виправлення вразливості.

1.4 Постановка завдання

У першому розділі проаналізовано структуру веб-додатків, можливу архітектуру та підходи до побудови веб-додатків. На основі проведеного аналізу необхідно зробити наступне:

- проаналізувати існуючі методи виявлення та ідентифікації вразливостей;
- розробити рішення для пошуку та ідентифікації вразливості типу LFI.

Висновки за розділом 1

У розділі 1 було наведено нормативно-правову базу у сфері захисту інформації, проаналізовано структуру сучасних веб-додатків, проведено аналіз архітектур веб-додатків та найбільш поширених підходів до їх побудови, розглянуто основні вразливості, що присутні веб-додаткам, причини їх появи та шляхи виправлення.

Веб-додатки проектуються за допомогою однієї із чотирьох архітектурних моделей: один веб-сервер і одна база даних, декілька веб-серверів і одна база даних, декілька веб-серверів і декілька баз даних, мікросервіс. Перші три моделі можна класифікувати як монолітну архітектуру. Остання є мікросервісною архітектурою.

Монолітна архітектура – це архітектура, в якій вся логіка сконцентрована в одній програмі. Перевагою даної архітектури є простота розробки і підтримки. Недоліком є висока залежність компонентів веб-додатку одне від одного, що ускладнює оновлення використовуваних технологій і зменшує надійність веб-додатку.

Мікросервісна архітектура – це архітектура, в якій логіка рознесена по різним модулям (сервісам), кожен з яких є незалежним одне від одного. Перевагою є висока

надійність даного підходу, відносна легкість оновлення окремих компонентів та технологій, що використовуються у системі. Недоліком є складність розробки.

У підрозділі 1.3 розглянуто та проаналізовано основні вразливості веб-додатків. У підрозділі 1.3.1 розглянуто топ-10 вразливостей веб-додатків за версією OWASP. У підрозділі 1.3.2 розглянуто топ-25 вразливостей програмного забезпечення за версією CWE та компанії MITRE. Також розглянуто CVE – список конкретних вразливостей певного програмного забезпечення.

В результаті аналізу можна зробити висновок, що найбільш поширеними вразливостями є:

- ін’єкції (OS Command, SQL, XSS);
- порушена аутентифікація;
- викриття чутливої інформації;
- зовнішні сутності XML (XML External Entities);
- порушення контролю доступу;
- неналежні конфігурації безпеки;
- проблеми десеріалізації.

РОЗДІЛ 2
МЕТОДИ ВИЯВЛЕННЯ ТА ІДЕНТИФІКАЦІЇ ВРАЗЛИВОСТЕЙ ВЕБ-ДОДАТКІВ

Зазвичай для пошуку вразливостей веб-додатків використовують спеціалізоване програмне забезпечення – інструменти оцінки вразливостей (vulnerability assessment tools). Прикладами можуть бути такі програми: nmap, Acunetix, Nessus, OpenVAS, Burp Suite, Nikto, NetSparker та інші [18].

Методи пошуку вразливостей, і, відповідно, інструменти, можна поділити на три групи [19]:

- статичні – ті, що перевіряють вихідний код додатків;
- динамічні – ті, що шукають вразливості у запущених додатках без доступу до вихідного коду;
- гібридні – ті, які поєднують статичні та динамічні методи.

2.1 Статичні методи

Статичне тестування безпеки додатків (static application security testing, SAST) – сукупність методів та технологій аналізу вихідного коду додатку з метою виявлення вразливостей [20]. Відноситься до «white-box testing», тобто вимагає доступу до внутрішніх структур додатку. Під час розробки додатків зазвичай використовується на стадіях програмування та тестування створених додатків.

Сканування за допомогою таких методів базується на сукупності заздалегідь визначених правил, які описують помилки в вихідному коді, на які слід звернути увагу. Такі методи здатні виявляти більшість вразливостей, починаючи від SQL-ін’єкцій та валідації введених даних до переповнення буферу.

Можна виділити наступні переваги:

- інтегрування тестування безпеки на ранні стадії розробки додатків, коли ще не нанесено жодної шкоди і виправлення помилок є простою задачею;
- забезпечення принципів secure coding – допомагає команді розробників притримуватись кращих практик та принципів розробки;
- високий рівень надійності при виявленні поширених вразливостей;
- повністю автоматизований метод пошуку за допомогою інструментів оцінки вразливостей;
- легкість виправлення вразливостей, оскільки буде знайдено точне місце в вихідному коді, де вона виникає.

Основними недоліками є:

- неможливість використання для вже створених та запущених додатків під час виконання;
- невичерпний список правил для проведення тестування;
- для великих додатків тестування займає багато часу;
- висока ймовірність отримання хибно позитивних результатів;
- складність інтеграції при використанні деяких фреймворків розробки (наприклад, Agile, при якому тестування коду додатку на ранніх етапах призведе до появи великої кількості знайдених вразливостей, оскільки розробники фокусують увагу на швидкості розробки, а не якості).

Розглянемо декілька прикладів використання статичних методів аналізу:

Приклад 1. SQL-ін’єкція.

Припустимо, що в коді додатку використовується наступний код (приклад взято з джерела [21]):

*String query = "SELECT * FROM accounts WHERE custID='" + request.getParameter("id") + "'";* (2.1)

В цьому випадку вразливість виникне через використання методу *request.getParameter("id")* і поєднання результату із запитом. Оскільки відсутні будь-які

перевірки вмісту параметру *"id"*, то зловмисник може підставити в цей параметр що завгодно і виконати певний запит, який не має мати змоги виконати.

Наприклад, в параметр *"id"* буде ін'єктовано «‘ or ‘1’ = ‘1». Тоді запит матиме наступний вигляд:

*String query = "SELECT * FROM accounts WHERE custID=" or '1'='1'";* (2.2)

Такий запит поверне всю таблицю accounts, навіть якщо атакуючий не має до неї прямого доступу.

Іншим можливим шляхом атаки, що захоче використати зловмисник може бути використання додаткового запиту. Для цього в параметр *"id"* можна ін'єктувати подібну конструкцію «’ ; ЗАПИТ». Таким чином атакуючий може виконати будь-яку дію, на яку спроможна мова запитів SQL.

Статичне тестування легко знайде таку вразливість, тому розробники додатку зможуть виправити її ще до того, як зловмисник отримає нагоду скористуватись нею.

Приклад 2. Використання вбудованих паролів.

Припустимо, що код додатку має наступний вигляд (приклад взято з джерела [22]):

```
#include <stdio.h>
int main()
{
  char *pwd = "super_password_not_that_secure";
  return 0;
}
```

Як легко побачити, в коді у змінній char *pwd вбудовано пароль «super_password_not_that_secure», який, на щастя не використовується в цій програмі, але моделює цілком реальну ситуацію.

Використання цього паролю в подальшому може призвести до порушення контролю доступу і отримання зловмисником доступу до тих частин додатку, з яких він зможе виконати інші зловмисні дії.

2.2 Динамічні методи

На відміну від статичних методів, динамічні (dynamic application security testing, DAST) методи застосовуються не для вихідного коду додатків, а для вже запущених додатків [23]. Відноситься до «black-box testing», тобто проводиться без доступу до внутрішніх структур та вихідного коду додатку. Такі методи напрямлені на роботу з фронт-ендом веб-додатку через веб-інтерфейс.

Іншою відмінною рисою є те, що динамічне тестування проводиться не лише за допомогою автоматизованих інструментів, а й вручну, оскільки деякі вразливості (наприклад, race condition чи помилки бізнес-логіки) інструментальним шляхом знайти неможливо.

Ще однією відмінністю від статичних методів є те, що для виявлення вразливостей доводиться дійсно проводити атаки на додатки, а не лише робити припущення щодо наявності вразливостей. Сканери симулюють шкідливу активність шляхом атакування та зондування, що дозволяє провести доволі реалістичну симуляцію атаки.

Такий підхід має наступні переваги:

- незалежність від додатку – можна тестувати будь-які додатки незалежно від мови програмування, фреймворку та ін.
- знайдені вразливості можна одразу експлуатувати;
- не вимагає доступу до вихідного коду;
- можливість тривалого проведення сканування для постійного пошуку нових вразливостей;

Існують наступні недоліки:

- неможливо знайти де саме у вихідному коді проявляється вразливість;
- без наявності знань щодо безпеки та вразливостей результати тестування не будуть дуже корисними;
- тестування може займати великі проміжки часу, в тому числі часу спеціаліста, який вручну шукає вразливості;
- існує можливість перезапису даних або ін'єкції шкідливого коду у робочий веб-додаток, тому вимагають створення окремого тестового середовища;
- неможливо покрити весь вихідний код та весь додаток, оскільки не вся частина додатку доступна «ззовні».

Конкретні техніки пошуку вразливостей залежать від самих вразливостей, на наявність яких і перевіряється веб-додаток. Наприклад, для пошуку SQL-ін'єкцій при динамічному тестуванні використовують програми по типу sqlmap або тестують усі можливі поля для вводу власноруч; для перевірки чи використовуються застарілі версії програмного забезпечення використовують nmap.

Таке програмне забезпечення зазвичай створюється для автоматизації кроків, які потрібні для експлуатування вразливостей: спочатку виконується аналіз «слідів» для визначення які сервіси чи програми використовуються, їх версії та наявність оновлень. Далі інструменти намагаються знайти індикатори того, що сервіси чи ПЗ є вразливими.

Інструменти оцінки вразливостей можна поділити на два види: активні та пасивні. Пасивні лише збирають дані щодо цілі і не ін'єктують нічого у ціль. В деяких випадках використовуються показники вразливості, тобто атрибути, які асоціюються з наявною вразливістю (наприклад, відсутність оновлення, яке виправляє певну вразливість). Такі інструменти обмежені у корисності (у порівнянні з активними), оскільки не тестують додатки на вразливості безпосередньо.

Активні (їх часто називають сканерами) реалізують певні техніки сканування, які включають у себе пошук ймовірних вразливостей (часто шляхом пасивного сканування) та ін'єктування певних даних у ту частину веб-додатку, де ймовірно існує

вразливість і дивляться на результати. Такі інструменти теж є обмеженими у корисності, оскільки працюють не знаючи деталей щодо цілі чи вразливостей.

Тому більшість інструментів пошуку вразливостей включають в себе як пасивні, так і активні техніки: пасивне сканування використовується для пошуку вразливостей, які ймовірно присутні у цільовому додатку, а активне сканування для підтвердження наявності знайдених вразливостей.

2.2.1 Пасивні

Розглянемо наступні приклади використання пасивних методів:

Приклад 1. Пошук технологій, використаних у веб-додатку.

Розуміння того, які технології та фреймворки використано у веб-додатку можуть допомогти зрозуміти його архітектуру та логіку. Зазвичай для цього використовують спеціалізоване програмне забезпечення, часто у вигляді додатків до веб-браузерів. Одним з таких додатків є Wappalyzer [24]. Приклад результатів роботи зображено на рисунку 2.1:

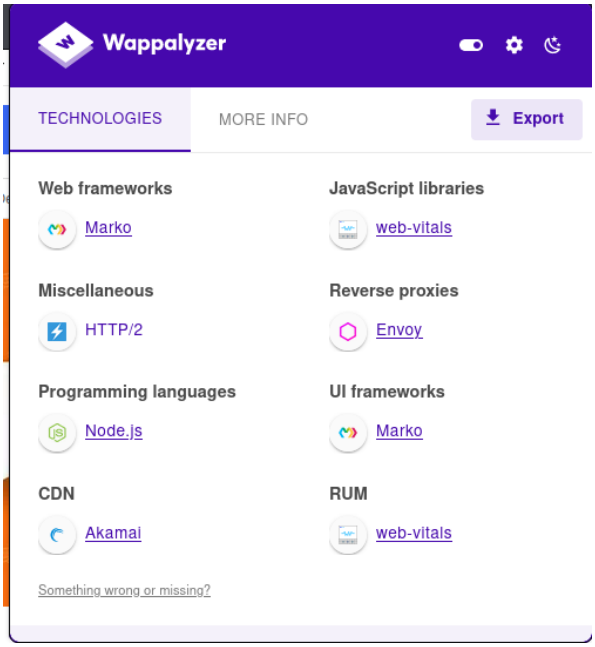

Рисунок 2.1. Приклад роботи додатку Wappalyzer.

В результаті швидкого сканування можна зрозуміти які фреймворки, мови програмування та бібліотеки використані у веб-додатку. Додатково можна дізнатись чи використовується мережа доставки контенту (Content Delivery Network, CDN). Завжди існує ймовірність того, що у веб-додатку використано небезпечну бібліотеку, для якої відомі вразливості і, відповідно, вразливим буде і сам додаток.

Приклад 2. Виявлення версії використовуваного додатку.

Така техніка пов’язана з вразливостями, що виникають внаслідок використання компонентів з відомими вразливостями. Незважаючи на зусилля розробників додатків, далеко не всі користувачі спішать оновлювати свої вразливі конфігурації за тих чи інших причин і це дуже часто призводить до компрометації системи.

Найчастіше для виявлення версій додатків використовують або ручний пошук версії на веб-сторінці або її вихідному коді, або програми по типу nmap для автоматизації процесу. Для окремих додатків використовується більш вузько спеціалізоване ПЗ. Наприклад, для пошуку версій модулів у WordPress використовується програма wpscan [25]. На рисунку 2.2 зображено процес виявлення версій використовуваного ПЗ за допомогою nmap [26]:

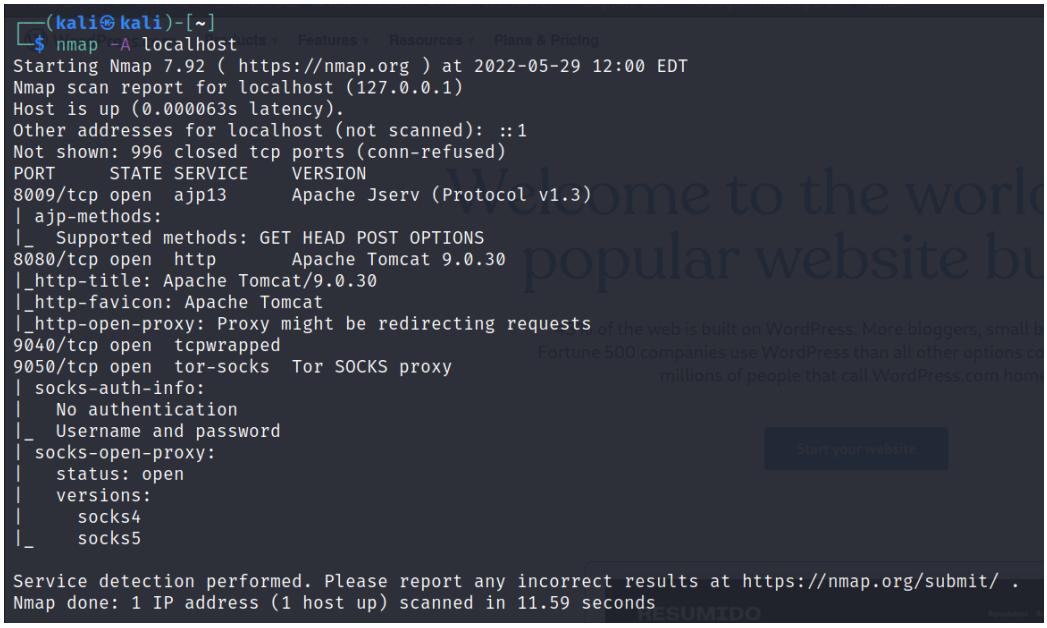

Рисунок 2.2. Пошук версій за допомогою утиліти nmap.

В цьому випадку було виявлено наступне:

- на веб-сервері відкриті порти 8009, 8080, 9040, 9050;
- на порті 8009 працює обробник Apache Jserv протоколу версії 1.3;
- на порті 8080 запущено Apache Tomcat версії 9.0.30;
- на портах 9040 та 9050 розгорнуто проксі з використанням Tor.

Можна зробити такі висновки: на сервері працює веб-сервер Apache Tomcat старої версії 9.0.30, для якої є відомі вразливості [27]; на сервері запущено проксі для маршрутизації вихідного трафіку через мережу Tor.

Пошук версій модулів WordPress з використанням утиліти wpscan зображено на рисунках 2.3 та 2.4 (URL- та IP- адреси приховано з метою збереження конфіденційності цільового додатку):

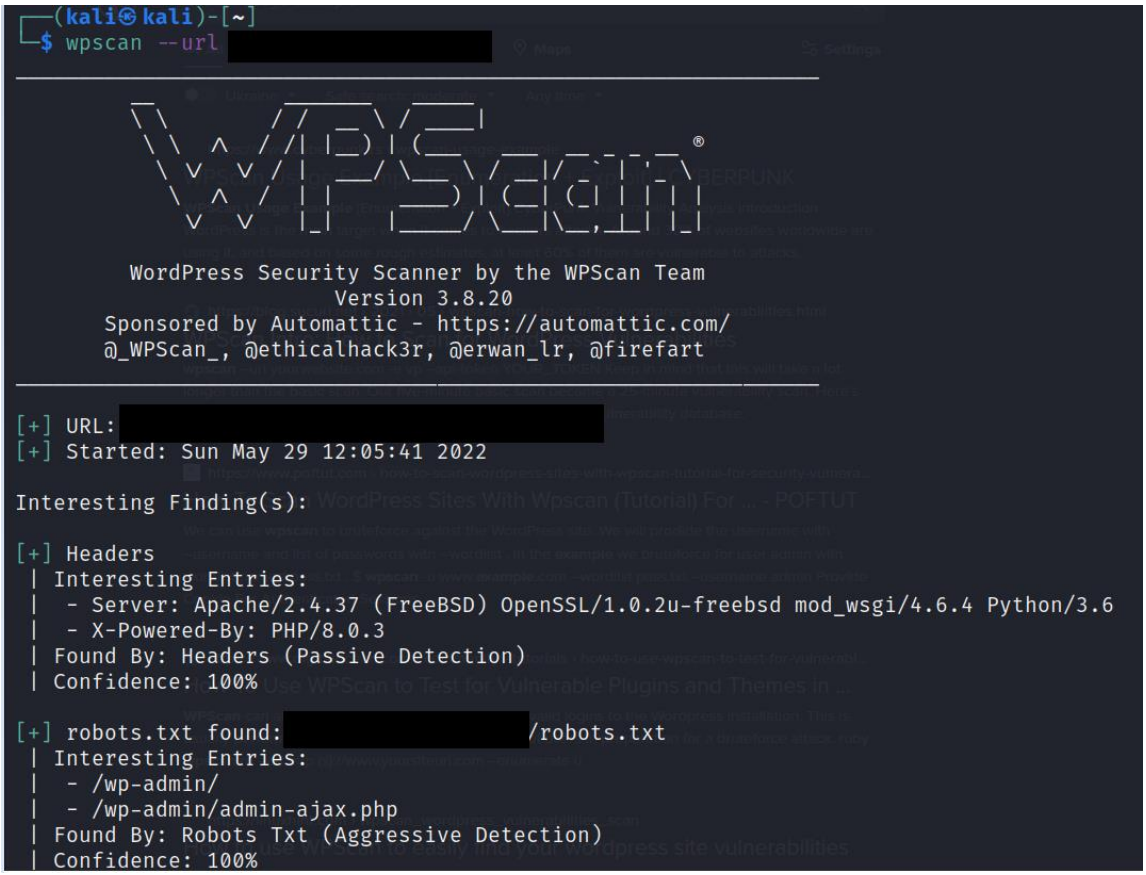

Рисунок 2.3. Пошук версій модулів WordPress за допомогою утиліти wpscan.

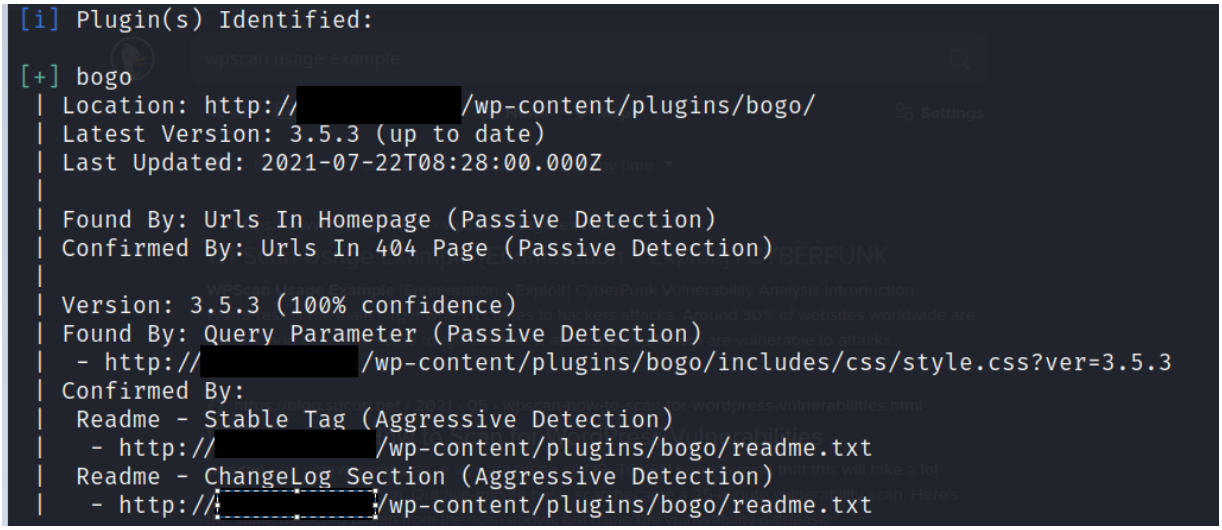

Рисунок 2.4. Пошук версій модулів WordPress за допомогою утиліти wpscan.

Як зображено на рисунках 2.3 та 2.4, утиліта wpscan виявила версію використаного веб-серверу – Apache 2.4.37, операційну систему FreeBSD та використані PHP та Python. Також було виявлено плагін bogo версії 3.5.3, яка на сьогодні є останньою.

Таким чином, пасивні методи виявлення вразливостей використовуються для збору інформації щодо веб-додатку і оцінки можливих місць появи вразливостей.

2.2.2 Активні

Розглянемо декілька прикладів:

Приклад 1. SQL-ін’єкція.

Припустимо, що потрібно перевірити наявність SQL-ін’єкції у певній частині додатку. У якості тестового додатку обрано лабораторію Portswigger для покращення навичок щодо SQL-ін’єкцій [28]. Існують два можливі шляхи виявлення SQL-ін’єкції у додатку – вручну та за допомогою автоматизованої утиліти (було обрано BurpSuite [29]).

На рисунку 2.5 зображено сторінку авторизації цільового додатку.

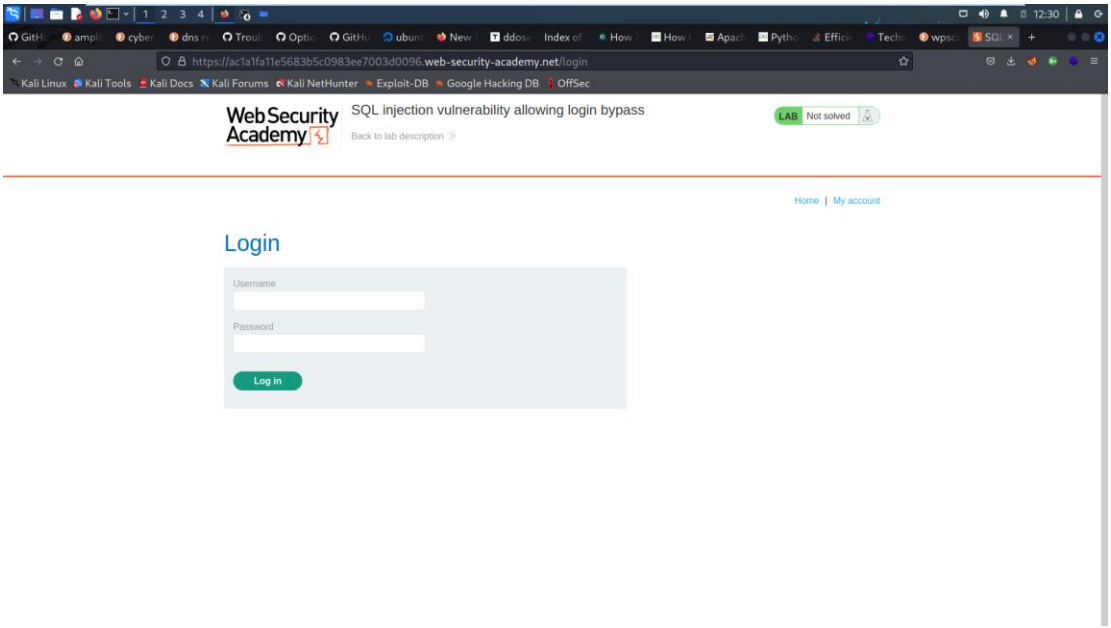

Рисунок 2.5. Сторінка авторизації.

Сторінка авторизації містить два поля вводу – username та password. Оскільки це навчальна лабораторія, то вже відомо що SQL-ін'єкція існує і пов'язана з логікою форми авторизації. Слід перехопити HTTP-запит авторизації та детально його розглянути. Для цього було використано програму Burp Suite, що зображено на рисунку 2.6:

```
POST /login HTTP/1.1
Host: ac1a1fa11e5683b5c0983ee7003d0096.web-security-academy.net
Cookie: session=VxeWMB97EjQdpQiyVwJSelSpzTdbQBTB
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:91.0) Gecko/20100101 Firefox/91.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded
Content-Length: 63
Origin: https://ac1a1fa11e5683b5c0983ee7003d0096.web-security-academy.net
Referer: https://ac1a1fa11e5683b5c0983ee7003d0096.web-security-academy.net/login
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: same-origin
Sec-Fetch-User: ?1
Sec-Gpc: 1
Te: trailers
Connection: close

csrf=Y4ec8WQXXc8WiV25KhUUXkr9dW0LoOPw&username=123&password=123
```

Рисунок 2.6. Перехоплений HTTP-запит.

Запит являє собою POST-запит, в якому параметри (ім’я користувача та пароль передаються у вигляді відкритому вигляді у тілі запиту). Скоріш за все, уразливим є або параметр username або password.

Якщо у параметр username підставити просту SQL-ін'єкцію вигляду administrator'--, то скоріш за все ін’єкцію буде успішно проведено і отримано доступ від імені користувача administrator. На рисунку 2.7 зображено змінений HTTP-запит.

```
POST /login HTTP/1.1
Host: ac1a1fa11e5683b5c0983ee7003d0096.web-security-academy.net
Cookie: session=VxeWMB97EjQdpQiyVwJSelSpzTdbQBTB
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:91.0) Gecko/20100101 Firefox/91.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded
Content-Length: 63
Origin: https://ac1a1fa11e5683b5c0983ee7003d0096.web-security-academy.net
Referer: https://ac1a1fa11e5683b5c0983ee7003d0096.web-security-academy.net/login
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: same-origin
Sec-Fetch-User: ?1
Sec-Gpc: 1
Te: trailers
Connection: close

csrf=Y4ec8WQXXc8WiV25KhUUXkr9dW0LoOPw&username=administrator'--&password=123
```

Рисунок 2.7. Змінений HTTP-запит.

Далі запит відправляється на сервер і повертається наступна веб-сторінка, яку зображено на рисунку 2.8:

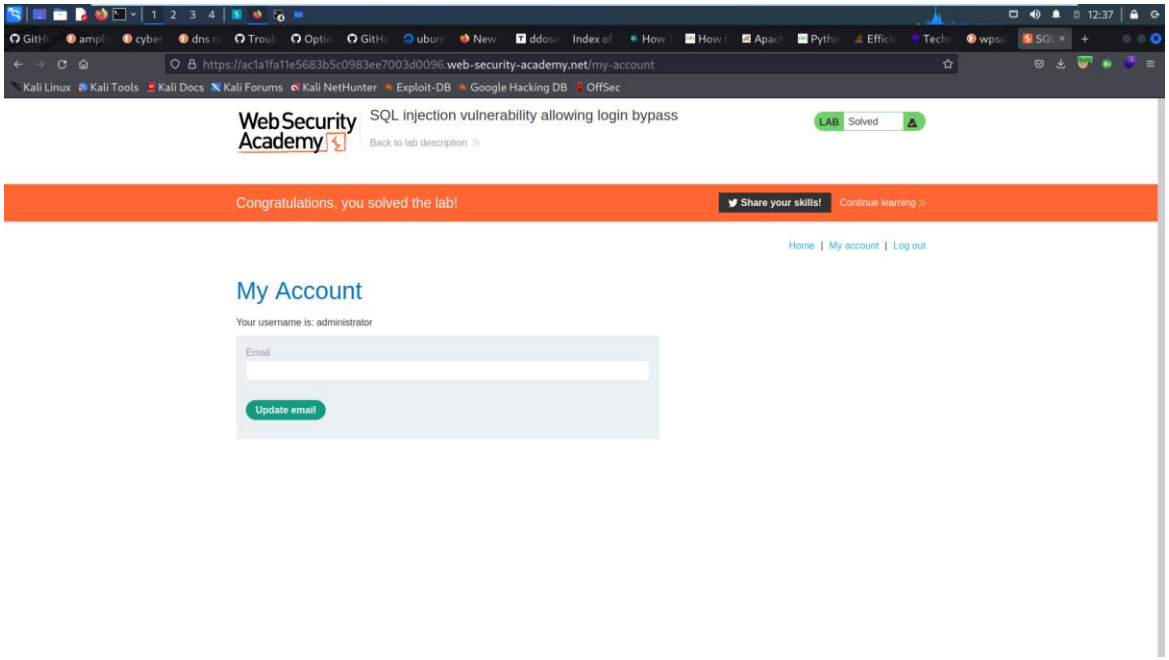

Рисунок 2.8. Відповідь від сервера.

Інший спосіб – провести ін’єкцію вручну. Для цього замість перехоплення запиту потрібно підставити ту саму SQL-ін'єкцію напряму у поле вводу username. Цей процес зображено на рисунку 2.9:

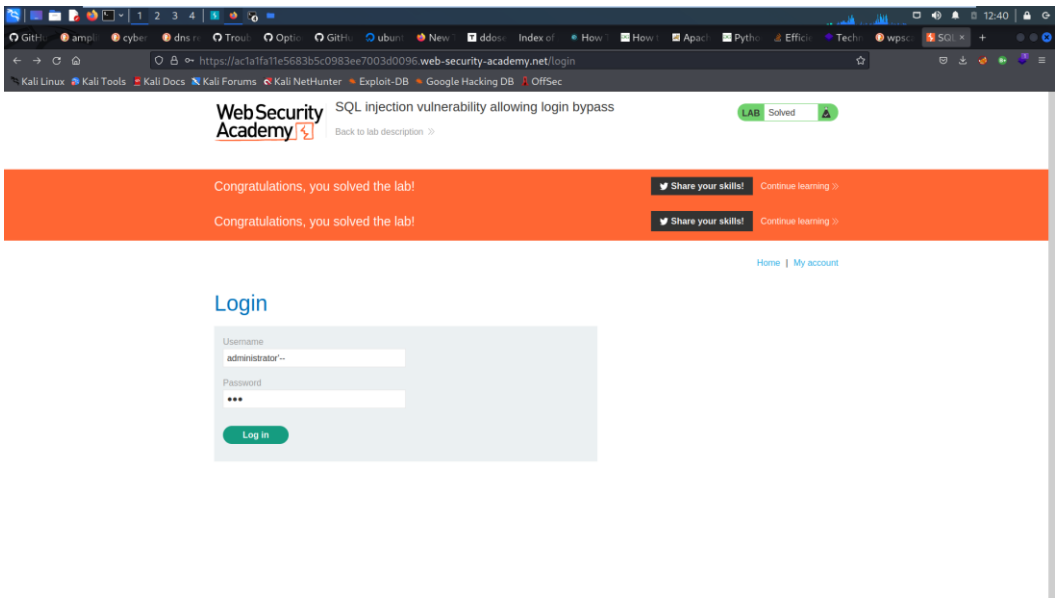

Рисунок 2.9. Проведення SQL-ін’єкції вручну.

Результат проведення ін’єкції буде такий самий, як і раніше – отримання доступу до веб-додатку від імені користувача administrator.

В цьому прикладі було проведено SQL-ін’єкцію, що дійсно впливає на роботу додатку. Можливо використати іншу SQL-ін’єкцію, яка не впливатиме на роботу додатку, але все одно покаже про наявність вразливості. Зазвичай у якості такої ін’єкції використовують ' UNION SELECT @@version--. Проведення такої SQL-ін’єкції зображено на рисунку 2.10. Результат ін’єкції зображено на рисунку 2.11.

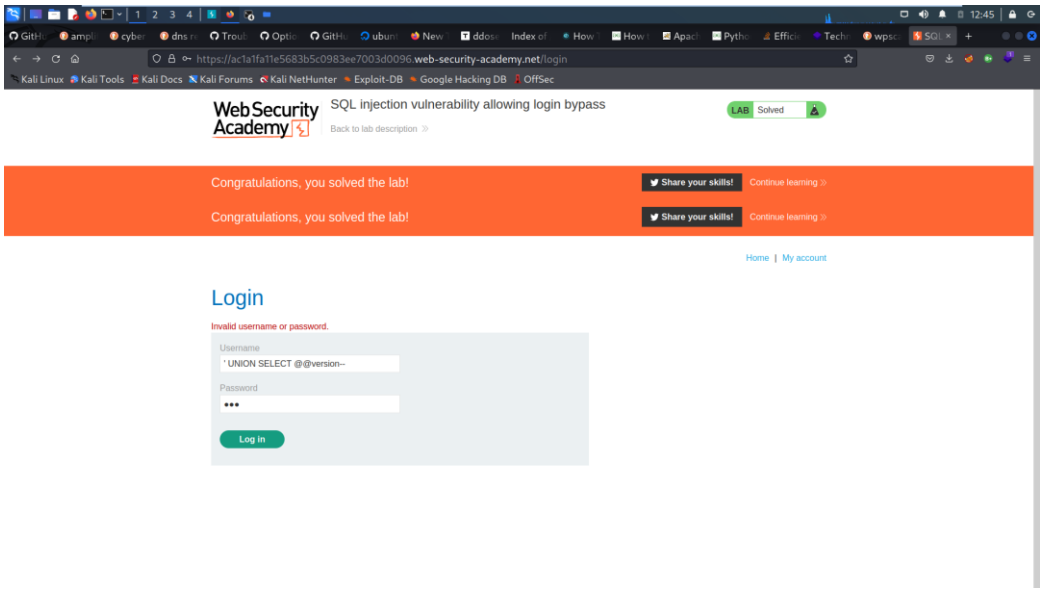

Рисунок 2.10. Проведення іншої SQL-ін’єкції.

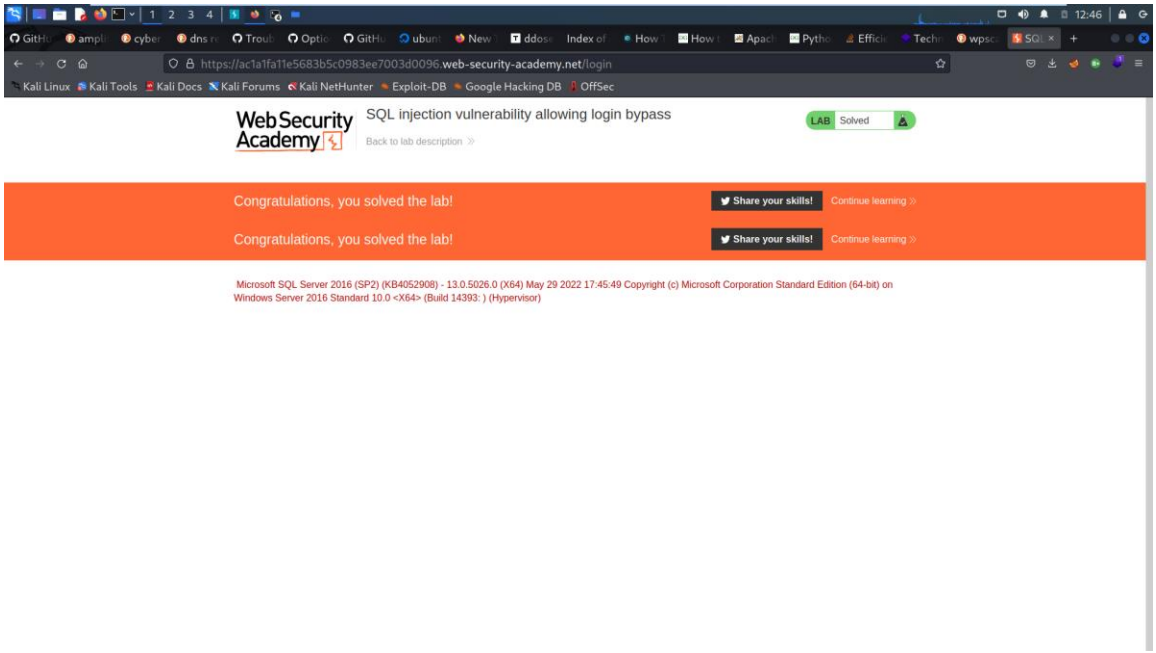

Рисунок 2.11. Результат проведення SQL-ін’єкції.

Як видно з рисунку вище, SQL-ін'єкція повернула в результаті версію SQL-серверу та операційної системи. Така ін’єкція дозволяє ідентифікувати вразливість не порушуючи роботу додатку.

Приклад 2. Виявлення вразливостей за допомогою спеціалізованого ПЗ.

Як вже неодноразово було вказано вище, зазвичай для пошуку вразливостей використовують програмне забезпечення – сканери вразливостей, яке дозволяє автоматизувати цей процес.

Одним з найкращих таких сканерів є Acunetix [30].

Нижче буде описано пошук вразливостей у веб-додатку за допомогою Acunetix.

У якості вразливого веб-додатку було обрано програму з відкритим вихідним кодом DIWA (Deliberately Insecure Web Application) [31] за авторством snsttr.

Основна сторінка веб-додатку зображена на рисунку 2.12:

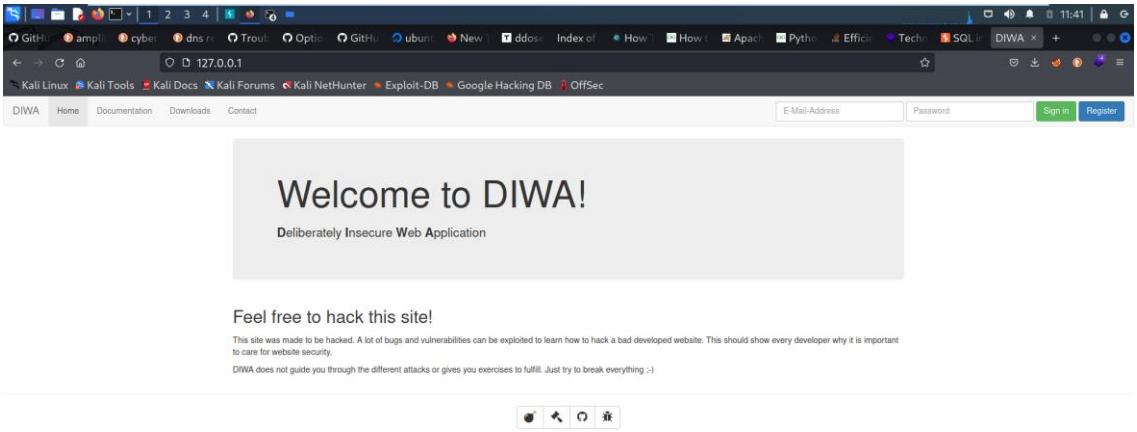

Рисунок 2.12. Головна сторінка веб-додатку DIWA.

Веб-додаток запущено на сервері з IP-адресою 192.168.142.128, веб-додаток прослуховує на порту 80. На рисунку 2.13 зображено пошук цієї адреси.

Рисунок 2.13. Визначення IP-адреси веб-серверу.

На іншому хості з ОС Windows запущено Acunetix, який прослуховує на порті 3443. Додання цілі до сканування зображено на рисунку 2.14.

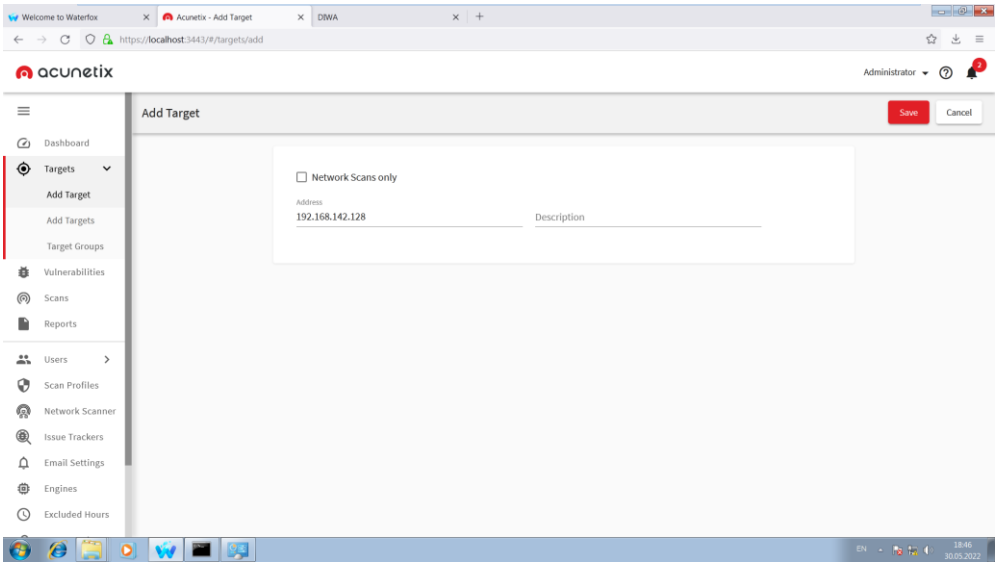

Рисунок 2.14. Додавання цілі до сканування.

Далі вносяться наступні параметри сканування:

- опис цілі;
- критичність цілі (більш критичні цілі перевіряються повільніше та з меншим навантаженням, не використовуються методи, які можуть призвести до порушення роботи);
- профіль сканування – можливість обрати певну вразливість чи повний спектр;
- швидкість сканування;
- додавання сторінки авторизації (хоча Acunetix здатен самостійно її виявляти);
- запис бізнес-логіки додатку;
- AcuSensor – додатковий модуль сканеру, який встановлюється окремо і покращує результати сканування зменшуючи кількість хибно позитивних результатів;
- параметри кроулінгу (пошуку прихованих директорій веб-серверу);
- імпорт параметрів сканування;
- параметри HTTP.

Вибір параметрів зображено на рисунках 2.15, 2.16 та 2.17. Незважаючи на велике розмаїття, для коректної роботи Acunetix достатньо просто вказати ціль і не змінювати інші параметри.

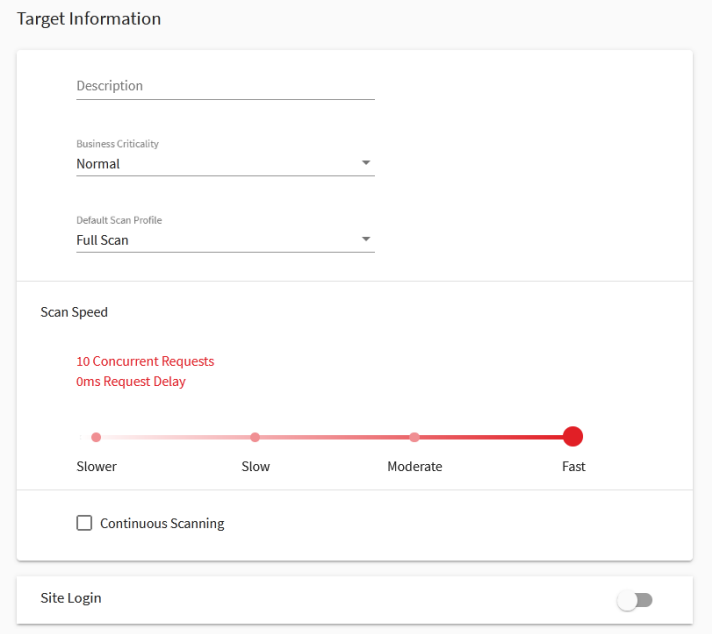

Рисунок 2.15. Вибір параметрів сканування.

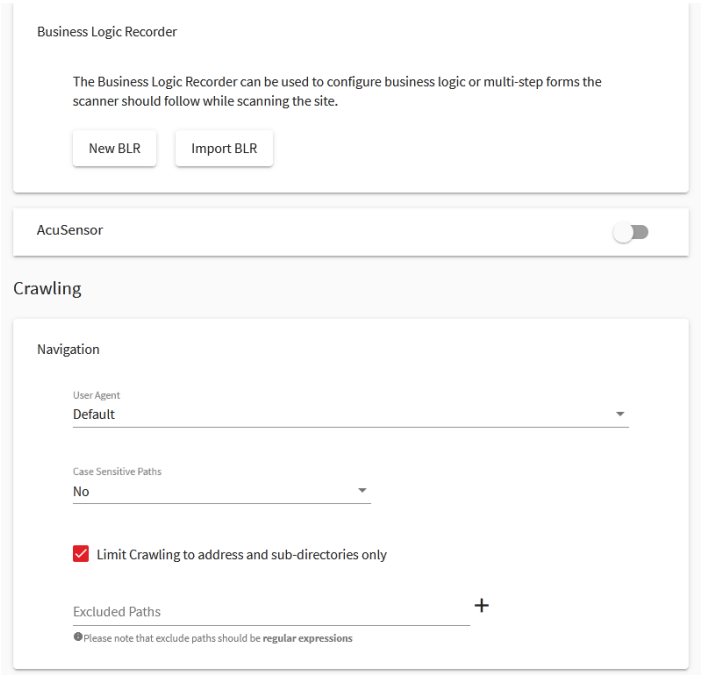

Рисунок 2.16. Вибір параметрів сканування.

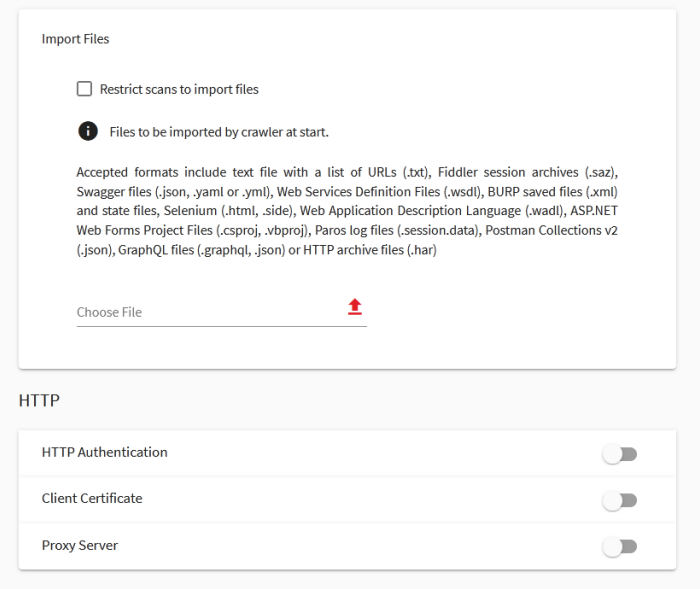

Рисунок 2.17. Вибір параметрів сканування.

Після вибору параметрів та запуску сканування відкривається дашборд програми, який висвітлює інформацію щодо сканування, в тому числі:

- витрачений час;
- кількість запитів;
- середній час відповіді;
- кількість знайдений шляхів (директорій) сканованого додатку;
- рівень загрози (ранжується від 0 до 3, 3 – найвищий);
- знайдені вразливості.

Сам дашборд зображено на рисунку 2.18.

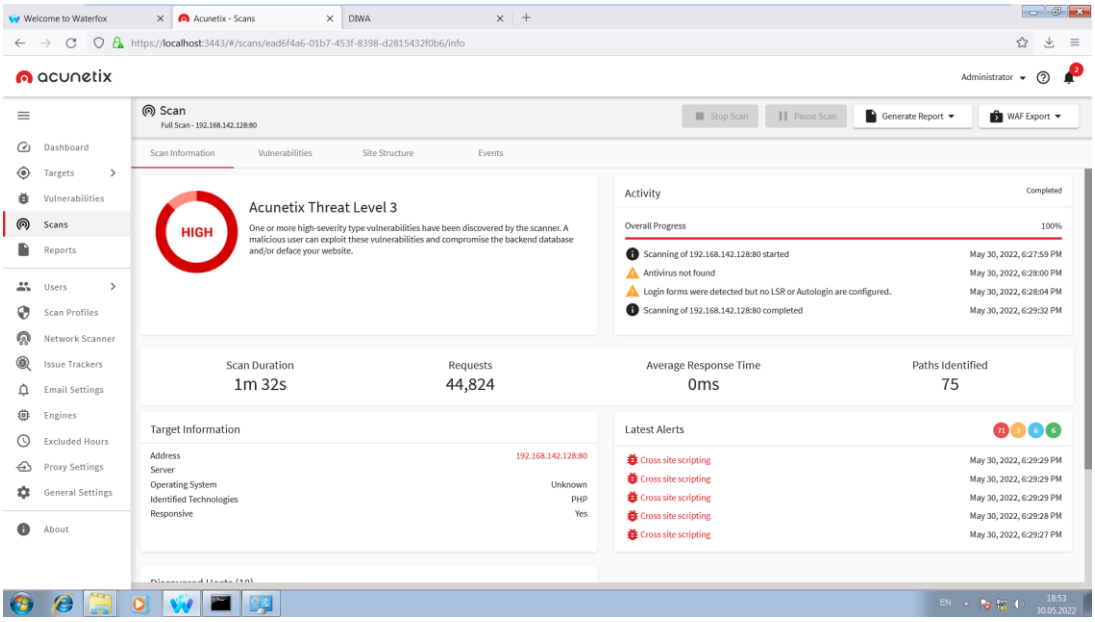

Рисунок. 2.18. Дашборд програми Acunetix після завершення сканування.

Далі користувач має змогу передивитись подробиці щодо кожної знайденої вразливості або згенерувати звіт. Приклади звітів щодо конкретних вразливостей зображено на рисунках 2.19 та 2.20. Такі звіти включають в себе інформацію щодо того, де саме було знайдено вразливість, який запит було використано, деталі атаки, опис вразливості та шляхи її виправлення. Приклад повного звіту включає себе подробиці кожної вразливості.

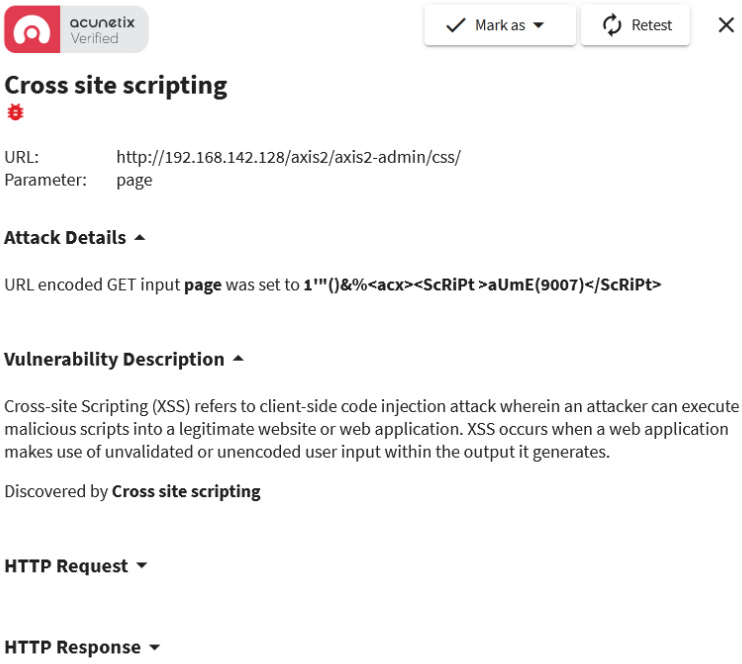

Рисунок 2.19. Звіт щодо знайденої вразливості XSS.

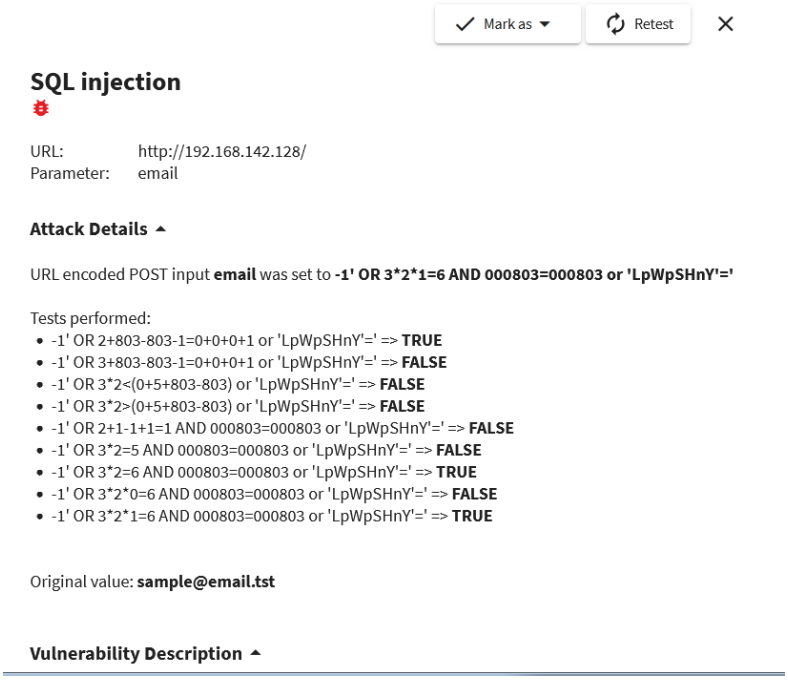

Рисунок 2.20. Звіт щодо знайденої вразливості SQL-ін’єкції.

Результати порівняння вищенаведених методів виявлення та ідентифікації вразливостей веб-додатків зображено у вигляді таблиці 2.1.

Таблиця 2.1

Порівняння методів пошуку вразливостей.

	Статичні	Динамічні	
		Пасивні	Активні
Тип тестування	White-box	Black-box	Black-box
Необхідний доступ	Вихідний код та внутрішні структури	Запущений веб-додаток	Запущений веб-додаток
Стадія життєвого циклу розробки ПЗ	Розробка та тестування	Тестування та використання	Тестування та використання
Легкість виправлення вразливостей	Легше	Складніше	Складніше
Можливості виявлення run-time вразливостей	Відсутні	Відсутні	Так
Легкість використання	Не вимагає особливих знань	Потрібні знання в області кібербезпеки	Потрібні знання в області кібербезпеки
Точність	Велика кількість хибно позитивних результатів	Залежить від рівня навичок спеціаліста, зазвичай висока	Залежить від рівня навичок спеціаліста, зазвичай висока

продовження табл.2.1

Покриття	Повне покриття веб-додатку	Менша, можливість знайти лише відомі вразливості (CVE) та атрибути	Можливість знайти будь-які вразливості у тих частинах додатку, до яких є доступ. Менше за статичні.
Орієнтованість	На розробника	На спеціаліста з безпеки	На спеціаліста з безпеки
Взаємодія	Лише з вихідним кодом додатку	Збір інформації щодо веб-додатку	Проведення атаки на веб-додаток

Гібридні методи виявлення та ідентифікації вразливостей являють собою поєднання статичних та динамічних в тілі однієї програми і містять всі переваги, які вони надають. Недоліки статичних методів перекриваються перевагами динамічних і навпаки, тому гібридні методи є найбільш ефективними. На жаль, наразі інструментів гібридного методу не так багато, оскільки вони потребують переосмислення існуючих інструментів і колосальних затрат спеціалістів.

Висновки за розділом 2

У розділі 2 розглянуто методи виявлення та ідентифікації веб-вразливостей.

Було проаналізовано наступні методи: статичні, динамічні, гібридні.

Статичні методи – сукупність методів та технологій аналізу вихідного коду додатку з метою виявлення вразливостей. Вони потребують обов’язкового доступу до вихідного коду додатків і перевіряють код за визначеним списком правил. Здатні виявляти широкий спектр вразливостей в усьому веб-додатку. Виправлення проблем

при використанні таких методів є доволі простим у виконанні, оскільки ці методи використовують під час розробки та первинного тестування додатків. Орієнтовані на розробників додатків і не потребують особливих знань у сфері кібербезпеки. Головним недоліком є велика кількість хибно позитивних результатів та невичерпність використовуваних правил.

Динамічні методи – сукупність методів та технологій тестування вже запущених додатків без доступу до вихідного коду програм. Здатні виявляти менший, порівняно зі статичними методами, спектр вразливостей, але при цьому здатні на виявлення тих проблем, які неможливо знайти за допомогою статичних методів, наприклад помилки бізнес-логіки або вразливості, що виникають лише під час роботи додатку. Потребують спеціаліста з кібербезпеки для проведення тестування, розуміння та інтерпретації його результатів. Такі методи не забезпечують повного покриття всього веб-додатку, а лише тих частин, до яких є зовнішній доступ. Виправлення помилок є важчим, ніж при використанні статичних методів, оскільки тестування відбувається лише при наявності вже розроблених додатків і не мають змоги виявити конкретне місце у вихідному коді, де виникає вразливість.

Динамічні методи в свою чергу можна поділити на пасивні та активні. Пасивні методи використовуються для збору інформації про додаток і потребують обмеженої взаємодії з додатком і не завжди спроможні однозначно виявити вразливість. Активні методи по суті симулюють проведення атак на веб-додаток і дозволяють однозначно перевірити чи наявна та чи інша вразливість у додатку. Поєднання двох методів є найбільш ефективним. Пасивні методи при такій техніці використання потрібні для виявлення місць, в яких можлива наявність вразливостей; активні для ідентифікації вразливостей.

Для порівняння вищеназваних методів було створено таблицю, в якій наочно показані переваги та недоліки цих методів.

Найбільш ефективними методами є гібридні методи – поєднання статичних та динамічних методів виявлення вразливостей. Їм присутні переваги обох типів, але

менша кількість недоліків, оскільки недоліки статичних методів виправляються динамічними і навпаки.

РОЗДІЛ 3
СТВОРЕННЯ СКАНЕРУ ВРАЗЛИВОСТЕЙ

Як було описано у розділі 2, деякі вразливості потребують створення сканерів, що орієнтовані на пошук саме цих вразливостей. Деякими прикладами таких є вразливостей CVE-2021-41773 [32], CVE-2021-42013 [33] та CVE-2020-1938 [34]. Всі вони пов’язані з продуктами компанії Apache і були активно експлуатовані зловмисниками у 2021 році. Дослідники називають ці вразливості одними з найбільш широко використовуваних. Незважаючи на те, що ці вразливості вже було виправлено за допомогою оновлень, велика кількість користувачів все одно використовує за тих чи інших причин вразливі конфігурації програмного забезпечення. Наразі немає зручного сканеру з відкритим вихідним кодом для цих вразливостей, який можна було б використовувати безкоштовно та без особливих знань у сфері кібербезпеки.

Оскільки розробити сканер вразливостей, що покрив би велику кількість вразливостей – задача, виконання якої потребує спільної роботи великої команди розробників, то краще почати з чогось меншого.

Тому при виконанні практичної частини був побудований додаток – сканер вищезгаданої вразливості CVE-2020-1938, що дозволяє виявити та ідентифікувати цю вразливість як статичними, так і динамічними методами. Мовою програмування було обрано Python версії 3.9.

3.1 Огляд вразливості CVE-2020-1938

Ця вразливість, яку пізніше було названо «GhostCat», була знайдена у січні 2020 року дослідниками компанії Chaitin Tech [35]. Вона існує у програмі Apache Tomcat – веб-додатку на базі мови Java. Вразливість пов’язана з реалізацією Apache JServ Protocol (AJP).

Tomcat використовує (у стандартній конфігурації) два порти: 8080 та 8009. Порт 8080 використовується для обробки HTTP-запитів, порт 8009 використовується службою AJP. AJP – бінарний протокол, який зменшує навантаження для серверу додатку. По принципу роботи він схожий на HTTP, але за рахунок бінарності набагато швидший.

Конектор AJP, який використовується у Tomcat, уражено вразливістю, використання якої дозволяє віддаленому неаутентифікованому зловмиснику отримати доступ до конфігурації і файлів вихідного коду додатків, які розгорнуто на веб-сервері. Якщо система при цьому дозволяє завантажувати файли на бік серверу (стандартна конфігурація), то зловмисник може завантажити та виконати довільний код за допомогою JavaServer Pages (JSP).

Типом вразливості є Path Traversal, що в деяких випадках може призвести до віддаленого виконання коду (RCE, Remote Code Execution). NIST визначає цю вразливість типом CWE-269 – improper privilege management [34][36].

Вразливість охоплює версії 6, 7, 8 та 9 використовуваного конектору при стандартній конфігурації, через що є критичною за рівнем ризику.

Вразливість була виправлена у лютому 2020 року компанією Apache шляхом оновлення Tomcat до версій 9.0.31, 8.5.51 та 7.0.100. Станом на травень 2022 року за допомогою системи Shodan.io [37] можна виявити тисячі систем, які не було оновлено до безпечних версій додатку, що зображено на рисунку 3.1.

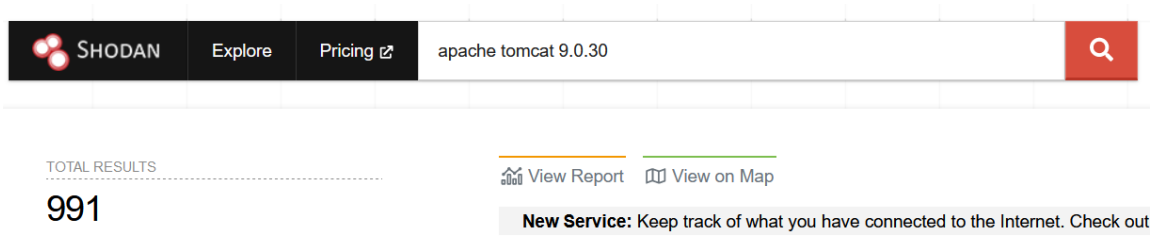

Рисунок 3.1. Результат пошуку вразливих версії додатку за допомогою сервісу Shodan.io.

У випадку використання динамічних методів для можливості наявності вразливості існують наступні умови:

- Tomcat запущено на сервері;
- використовується небезпечна (неоновлена) версія додатку;
- порт 8009 відкрито.

При використанні статичних методів для виявлення вразливості слід:

- переконатись, що встановлено небезпечну версію додатку;
- переконатись, що у конфігурації включено конектор AJP – у файлі server.xml наявна і не закоментована наступна строка коду:

<Connector port="8009" protocol="AJP/1.3" redirectPort="8443" />

3.2 Огляд створеного додатку

Створений додаток отримав назву «GhostCat scanner» версії 1.0.

При запуску додаток пропонує три можливі опції – статичний аналіз, динамічний аналіз і вихід. Головне меню зображено на рисунку 3.2.

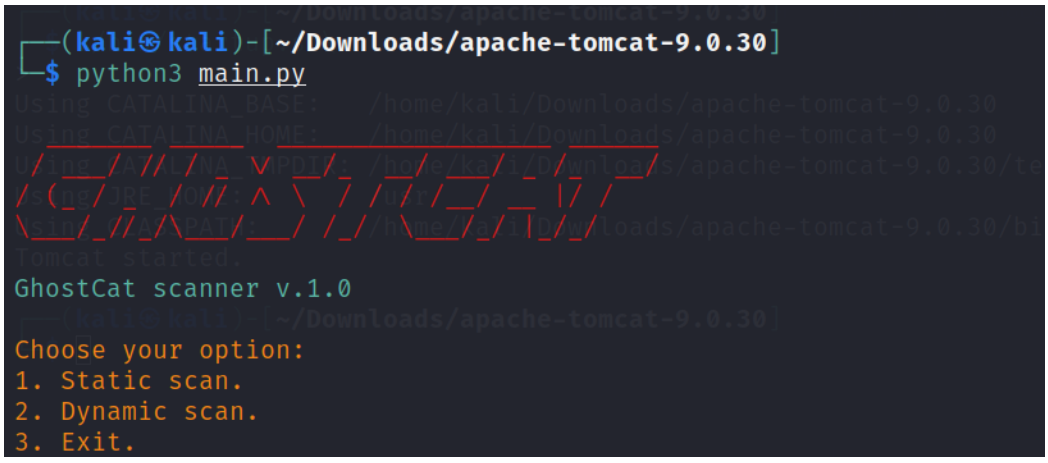

Рисунок 3.2. Головне меню програми

Додаток має два окремі функціональні блоки – блок для статичного аналізу локальної конфігурації та блок для динамічного аналізу додатку.

Блок для статичного аналізу виконує наступне:

1. Запитує повний шлях до директорії, в якій розгорнуто Tomcat (рисунок 3.3).

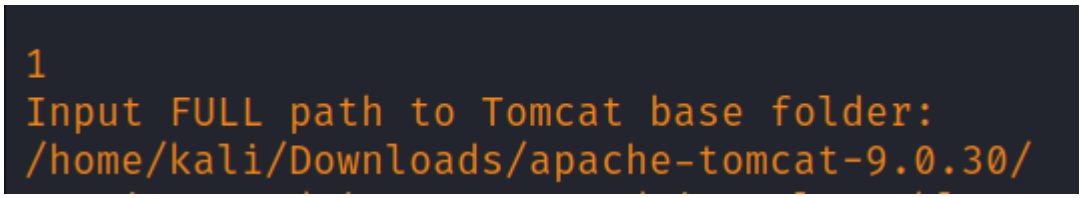

Рисунок 3.3. Ввід повного шляху.

2. Перевіряє версію Tomcat шляхом читання файлу RELEASE-NOTES, який йде разом з Tomcat при інсталяції і описує версію.

3. Перевіряє чи використовується AJP.

4. Якщо використано вразливу версію Tomcat та включено конектор AJP – конфігурація є вразливою (рисунок 3.4).

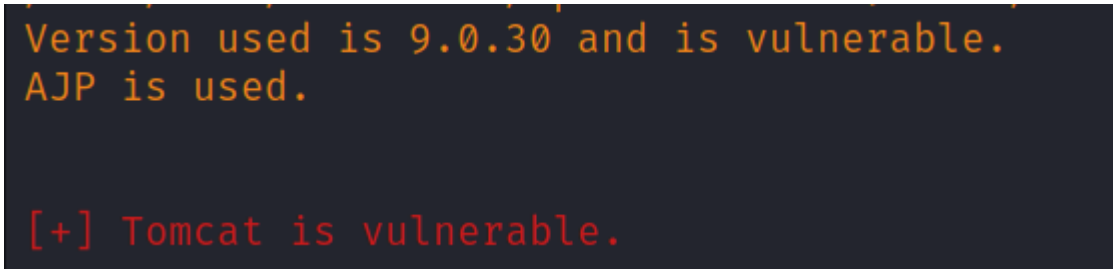

Рисунок 3.4. Результат аналізу вразливого додатку.

Якщо оновити Tomcat (в цьому випадку до версії 9.0.32), то результати аналізу будуть іншими. Приклад зображено на рисунку 3.5.

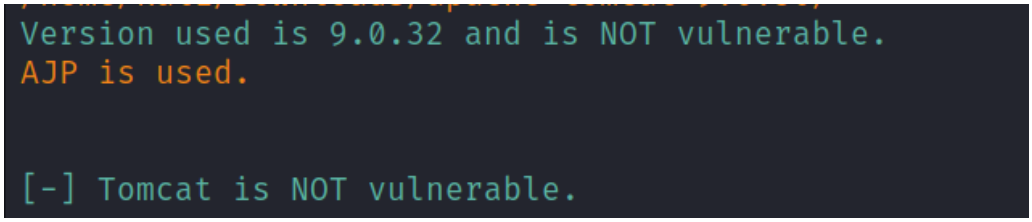

Рисунок 3.5. Результат аналізу невразливого додатку.

Блок динамічного аналізу виконує наступне:

1. Запитує IP-адресу цільового сервера (рисунок 3.6).

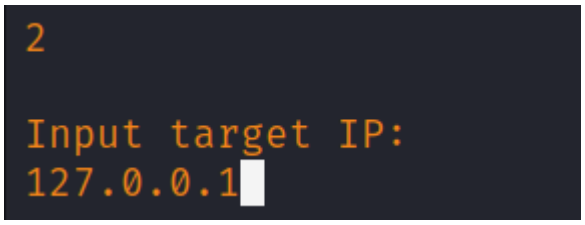

Рисунок 3.6. Ввід IP-адреси.

2. Запитує порт, на якому прослуховує Tomcat (рисунок 3.7).

Рисунок 3.7. Запит порту, на якому прослуховує Tomcat.

3. Запитує порт, на якому прослуховує служба конектора AJP (рисунок 3.8).

Рисунок 3.8. Запит порту, на якому прослуховує служба AJP.

4. За допомогою модулю python-nmap сканує сервер за вказаними параметрами:
    a. Перевіряється чи запущено Tomcat та його версія,
    b. Перевіряється чи запущено службу AJP.
5. Робить висновок щодо вразливості додатку (рисунок 3.9).

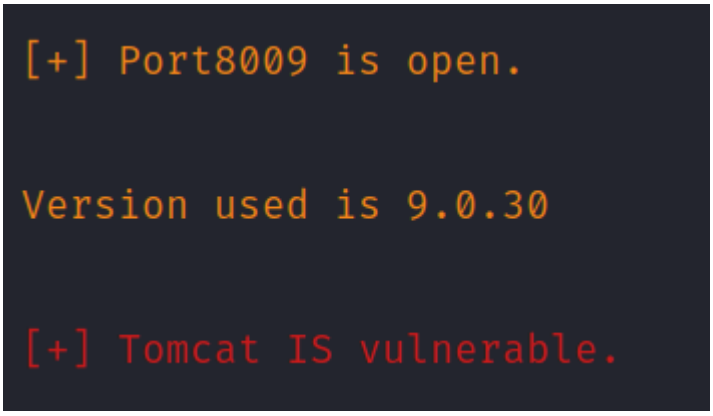

Рисунок 3.9. Результат сканування додатку.

3.3 Оцінка ефективності рішення

Розроблений додаток дозволяє однозначно визначити відсутність вразливості у цільовому додатку, але не надає таких гарантій щодо наявності вразливості.

Для того щоб додаток не був вразливим достатньо використовувати незастарілу версію додатку, або не використовувати службу AJP.

Для того щоб додаток був вразливим необхідно одночасне використання служби AJP та застарілої версії Tomcat. Однак існують такі конфігурації Tomcat, в якіх одночасно виконуються вищеназвані умови, але не існує можливості експлуатувати вразливість. Однією з таких конфігурацій є використання атрибуту requiredSecret у параметрах служби конектора AJP (рисунок 3.10). Це дозволяє обмежити доступ до конектора тим користувачам, які не мають цього секрету. Використання атрибуту requiredSecret по суті є аналогом пароля.

```
114
115     <!-- Define an AJP 1.3 Connector on port 8009 -->
116     <Connector protocol="AJP/1.3"
117                address="TOMCAT_IP_ADDRESS"
118                port="8009"
119                redirectPort="8443"
120                requiredSecret="YOUR_AJP_SECRET_GOES_HERE" />
121
```

Рисунок 3.10. Безпечна конфігурація.

Виявити чи використовується така конфігурація в існуючому стані розроблений сканер не здатен. Такий функціонал за потреби можна реалізувати, але це вимагатиме переробки логіки сканеру.

Висновки за розділом 3

У розділі 3 було розглянуто вразливість CVE-2020-1938 та побудовано додаток – сканер, що дозволяє виявити цю вразливість у цільовому додатку Tomcat. Сканер побудовано мовою Python.

Сканування було виконано двома різними методами аналізу вразливостей – статичного та динамічного. За допомогою статичного аналізу було проаналізовано використовуваний додаток Tomcat та зроблено висновок щодо вразливості. При використанні статичних методів сканер і Tomcat мають бути запущені на одному сервері. Динамічний аналіз дозволяє провести пошук вразливості у будь-якому віддаленому сервері.

Для виявлення вразливості використовуються підходи, що зумовлено особливостями вразливості – перевірка версії додатку Tomcat та перевірка використання служби Apache JServ Protocol.

Недоліком додатку є те, що ефективність додатку дещо обмежена, оскільки існують такі конфігурації додатку, які не є вразливими, але які побудований сканер не дозволяє визначити, помилково оцінюючи вразливості цільового додатку.

ВИСНОВКИ

У дипломній роботі було побудовано сканер вразливостей, що здатен виявляти вразливу до CVE-2020-1938 конфігурацію, веб-додатку Apache Tomcat. Сканер поєднує у собі статичні та динамічні методи виявлення вразливостей.

У першій частині дипломної роботи було проведено аналіз принципів роботи веб-додатків та основних найбільш поширених вразливостей веб-додатків. Аналіз включає у себе огляд нормативно-правової бази в галузі інформаційної безпеки, визначення поширених архітектур та структури веб-додатків, визначено найбільш поширені вразливості та методи їх виправлення відповідно до джерел [14] та [15]. Цими вразливостями є:

- ін’єкції (OS Command, SQL, XSS);
- порушена аутентифікація;
- викриття чутливої інформації;
- зовнішні сутності XML (XML External Entities);
- порушення контролю доступу;
- неналежні конфігурації безпеки;
- проблеми десеріалізації.

У другій частині дипломної роботи було розглянуто методи виявлення та ідентифікації вразливостей веб-додатків. Цими методами є статичний аналіз, динамічний аналіз та гібридний аналіз, кожен з яких має свої переваги та недоліки. На конкретних прикладах розглянуто кожен з методів та його особливості. Проведено порівняння вищевказаних методів.

У третій частині роботи на основі аналізу, проведеного у першій та другій частинах було побудовано сканер для виявлення вразливості CVE-2020-1938.

Вразливість виникає у стандартних конфігураціях веб-додатку Apache Tomcat і дозволяє неаутентифікованим користувачам переглядати будь-які файли на сервері, де

запущено вразливий додаток і віддалено виконувати будь-який власний код, що робить цю вразливість дуже небезпечною, більшою мірою завдяки тому, що вона виникає саме у стандартних конфігураціях. Не зважаючи на те, що вразливість було виправлено оновленнями, наразі не всі користувачі скористались виправленнями та існують тисячі вразливих систем.

Виявлення вразливості досягається завдяки перевірці обов’язкових для наявності вразливості налаштувань веб-додатку – використання старої версії та використання служби Apache JServ Protocol.

Виходячи із поставленої мети дипломної роботи були поставлені та виконані наступні завдання:

- досліджено структуру веб-додатків;
- проведено аналіз найбільш поширених вразливостей;
- проаналізовано існуючі методи виявлення та ідентифікації вразливостей;
- розроблено рішення для пошуку вразливості типу LFI.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Hackers Breached Colonial Pipeline Using Compromised Password [електронний ресурс]: Bloomberg. Режим доступу: https://www.bloomberg.com/news/articles/2021-06-04/hackers-breached-colonial-pipeline-using-compromised-password

2. U.S. declares state of emergency over Colonial pipeline shutdown [електронний ресурс]: CGTN. Режим доступу: https://news.cgtn.com/news/2021-05-10/U-S-declares-state-of-emergency-over-Colonial-pipeline-shutdown-108R5zDzXXi/index.html

3. Повідомляють про масштабний злив даних користувачів «Дія», Мінцифри це заперечує. [електронний ресурс]: DOU. Режим доступу: https://dou.ua/lenta/news/diia-data-leak-2022/

4. Інформація про витік даних із порталу "Дія" не відповідає дійсності. [електронний ресурс]: МВС України. Режим доступу: https://mvs.gov.ua/news/informaciya-pro-vitik-danix-iz-portalu-diya-ne-vidpovidaje-diisnosti

5. Про інформацію [Електронний ресурс]: Закон України № 2657-XII від 16.06.2020. – Режим доступу: https://zakon.rada.gov.ua/laws/show/2657-12#Text.

6. Про захист персональних даних [Електронний ресурс]: Закон ¬¬¬України № 2297-VI від 23.04.2021. – Режим доступу: https://zakon.rada.gov.ua/laws/show/2297-17#Text.

7. Про затвердження Положення про організацію заходів із забезпечення інформаційної безпеки в банківській системі України [Електронний ресурс]: Постанова Правління Національного банку України №95 від 28.09.2017. – Режим доступу: https://zakon.rada.gov.ua/laws/show/v0095500-17#Text .

8. NIST Special Publication 800-63B. Digital Identity Guidelines. Authentication and Lifecycle Management. [електронний ресурс]: NIST. – Режим доступу: https://pages.nist.gov/800-63-3/sp800-63b.html.

9. RFC 2616: Hypertext Transfer Protocol – HTTP/1.1.

10. ISO/IEC 27000:2018. Information technology – Security techniques – Information security management systems – Overview and vocabulary.

11. Best language for web application development. [електронний ресурс]: steelwiki. – Режим доступу: https://steelkiwi.com/blog/best-languages-for-web-application-development

12. Web application architecture: Components, models and types [електронний ресурс]: ScienceSoft. – Режим доступу: https://www.scnsoft.com/blog/web-application-architecture

13. НД ТЗІ 1.1-003-99: Термінологія в галузі захисту інформації в комп'ютерних системах від несанкціонованого доступу. Київ: Департамент спеціальних телекомунікаційних систем та захисту інформації Служби безпеки України. 1999.

14. OWASP Top Ten [електронний ресурс]: OWASP. – Режим доступу: https://owasp.org/www-project-top-ten.

15. Common Weakness Enumeration [електронний ресурс]: Mitre. – Режим доступу: https://cwe.mitre.org.

16. CVE [електронний ресурс]: Mitre. – Режим доступу: https://cve.mitre.org.

17. Top 25 software errors [електронний ресурс]: SANS. – Режим доступу: https://www.sans.org/top25-software-errors.

18. Web application security testing [електронний ресурс]: Getastra. – Режим доступу: https://www.getastra.com/blog/security-audit/web-application-security-testing

19. A Dose of Reality on Automated Static-Dynamic Hybrid Analysis [електронний ресурс]: Veracode. – Режим доступу: https://www.veracode.com/blog/secure-development/whitepaper-dose-reality-automated-static-dynamic-hybrid-analysis.

20. What is Static Application Security Testing? [електронний ресурс]: Microfocus. – Режим доступу: https://www.microfocus.com/en-us/what-is/sast.

21. A1:2017 Injection [електронний ресурс]: OWASP. – Режим доступу: https://owasp.org/www-project-top-ten/2017/A1_2017-Injection.

22. Hardcoded passwords [електронний ресурс]: The Security Vault. – Режим доступу: https://thesecurityvault.com/hardcoded-passwords.

23. What is Dynamic Application Security Testing? [електронний ресурс]: Microfocus. – Режим доступу: https://www.microfocus.com/en-us/what-is/dast.

24. Wappalyzer [електронний ресурс]: Wappalyzer. – Режим доступу: https://www.wappalyzer.com.

25. WPScan [електронний ресурс]: WPScan. – Режим доступу: https://wpscan.com.

26. Nmap: the network mapper [електронний ресурс]: Nmap. – Режим доступу: https://nmap.org.

27. Apache Tomcat Security Vulnerabilities Published in 2020 [електронний ресурс]: CVE Details. – Режим доступу: https://www.cvedetails.com/vulnerability-list.php?vendor_id=45&product_id=887&version_id=644763&page=1&hasexp=0&opdos=0&opec=0&opov=0&opcsrf=0&opgpriv=0&opsqli=0&opxss=0&opdirt=0&opmemc=0&ophttprs=0&opbyp=0&opfileinc=0&opginf=0&cvssscoremin=0&cvssscoremax=0&year=2020&cweid=0&order=1&trc=1&sha=2a8dd49fb6c2bfb42e5b75235217cc1195a4f972

28. Lab: SQL Vulnerability allowing login bypass [електронний ресурс]: PortSwigger. – Режим доступу: https://portswigger.net/web-security/sql-injection/lab-login-bypass.

29. Burp Suite [електронний ресурс]: PortSwigger. – Режим доступу: https://portswigger.net/burp.

30. Acunetix [електронний ресурс]: Invicti. – Режим доступу: https://www.acunetix.com.

31. Deliberately Insecure Web Application by snsttr [електронний ресурс]: GitHub. – Режим доступу: https://github.com/snsttr/diwa.

32. CVE-2021-41773 [електронний ресурс]: National Vulnerability Database. – Режим доступу: https://nvd.nist.gov/vuln/detail/CVE-2021-41773.

33. CVE-2021-42013 [електронний ресурс]: National Vulnerability Database. – Режим доступу: https://nvd.nist.gov/vuln/detail/CVE-2021-42013.

34. CVE-2020-1938 [електронний ресурс]: National Vulnerability Database. – Режим доступу: https://nvd.nist.gov/vuln/detail/CVE-2020-1938.

35. Busting Ghostcat: Analysis of CVE-2020-1938 [електронний ресурс]: TrendMicro. – Режим доступу: https://www.trendmicro.com/en_us/research/20/c/busting-ghostcat-an-analysis-of-the-apache-tomcat-vulnerability-cve-2020-1938-and-cnvd-2020-10487.html.

36. CWE-269: Improper Privilege Management [електронний ресурс]: MITRE. – Режим доступу: https://cwe.mitre.org/data/definitions/269.html

37. Search Engine for the Internet of Everything [електронний ресурс]: Shodan. – Режим доступу: https://www.shodan.io.

ДОДАТОК А

Вихідний код розробленого додатку

```
import re
import nmap

# terminal colors
red = '\033[31m'
green = '\033[32m'
yellow = '\033[93m'


def version_check(version):  # check if version is vulnerable
    if version[0] == "6":  # tomcat6
        return True
    if version[0] == "7":  # tomcat7
        if "0" <= version[2] <= "100":
            return True
    if version[0] == "8":  # tomcat8
        if "0" <= version[1] <= "4":
            return True
        if version[1] == "5" and "0" <= version[2] <= "51":
            return True
    if version[0] == "9":  # tomcat9
        if version[1] == "0" and "0" <= version[2] < "31":
            return True
```

```
def static_check_ver(base_path):  # check for version in RELEASE-NOTES file
    target = base_path + "RELEASE-NOTES"
    with open(target, 'r') as f:
        version_lines = [line for line in f if "Version" in line]  # parsing version
    version = re.findall('[0-9]+', version_lines[1])
    if version_check(version):
        print(yellow + "Version used is " + version[0] + "." + version[1] + "." + version[2]
+ " and is vulnerable.")
        return True
    else:
        print(green + "Version used is " + version[0] + "." + version[1] + "." + version[2]
+ " and is NOT vulnerable.")
        return False


def static_check_ajp(base_path):  # check for AJP usage
    target = base_path + "/conf/server.xml"
    ajp_string = "<Connector port=\"8009\" protocol=\"AJP/1.3\" redirectPort=\"8443\"
/>"
    with open(target, 'r') as f:
        ajp_lines = [line for line in f if ajp_string in line]  # parsing version
    if ajp_lines and "!--" not in ajp_lines:
        print(yellow + "AJP is used.")
        return True
    else:
        print(green + "AJP is NOT used.")
        return False
```

```
def dynamic_version(host, port):  # find Tomcat version using nmap
    nm = nmap.PortScanner()
    nm.scan(host, port)
    port_key = int(port)
    ver = nm[host]['tcp'][port_key]['version']
    print("\nVersion used is " + ver)
    version = re.findall(r"\d+", ver)
    return version


def dynamic_port(host, port):
    nm = nmap.PortScanner()
    scanned = nm.scan(host, port)
    port_key = int(port)
    status = scanned['scan'][host]['tcp'][port_key]['state']
    if status == "open":
        print("\n[+] Port" + port + " is open.\n")
        return True
    else:
        return False


def static_method():  # static check for local installation

    tomcat_base = str(input("Input FULL path to Tomcat base folder:\n"))
    try:
```

```
        static_check_ver_res = static_check_ver(tomcat_base)
        static_check_ajp_res = static_check_ajp(tomcat_base)
        if static_check_ver_res is True and static_check_ajp_res is True:
            print(red + "\n\n[+] Tomcat is vulnerable.")
        else:
            print(green + "\n\n[-] Tomcat is NOT vulnerable.")
    except:
        print(red + "Incorrect folder provided.\nExiting.")
        exit()


def dynamic_method():  # dynamic check for remote host
    host = str(input("\nInput target IP:\n"))
    try:
        tomcat_port = int(input("Input target Tomcat port (default: 8080) :\n"))
    except:
        print("\nIncorrect Tomcat port provided. Resetting to default\n")
        tomcat_port = 8080
    tomcat_port = str(tomcat_port)

    try:
        ajp_port = int(input("\nInput target AJP port (default: 8009) :\n"))
    except:
        print("\nIncorrect AJP port provided. Resetting to default\n")
        ajp_port = 8009
    ajp_port = str(ajp_port)

    port_check = dynamic_port(host, ajp_port)  # check if port 8009 is open
```

```
    if not port_check:
        print(green + "\nPort 8009 closed, Tomcat is NOT vulnerable.\n\n")
        exit()
    version_vulnerable = version_check(dynamic_version(host, tomcat_port))  # find version & check if vulnerable
    if port_check is True and version_vulnerable is True:
        print(red + "\n\n[+] Tomcat IS vulnerable.\n")
    else:
        print(green + "\n\n[-] Tomcat is NOT vulnerable.")



def banner():

    banner_logo = """
  _______ _____  ____________________ ______
 / ___/ // / _ \/ __/_  __/ ___/ _ /_  __/
/ (_ / _  / // /\ \  / / / /__/ __ |/ /
\___/_//_/\___/___/ /_/  \___/_/ |_/_/    \n
"""
    print(red + banner_logo + green + "GhostCat scanner v.1.0\n\n" + yellow + "Choose your option:\n1. Static scan.\n2. Dynamic scan."
                                                "\n3. Exit.\n")

if __name__ == '__main__':
    banner()
    method = str(input())
    if method == '1':
        static_method()
```

```
    elif method == '2':
        dynamic_method()
    else:
        print("Unsupported option.")
exit()
```