Міністерство освіти і науки України
Харківський національний університет імені В.Н. Каразіна
Навчально-науковий інститут комп'ютерних наук та штучного інтелекту
Спеціальність 125 «Кібербезпека»
Освітня програма «Кібербезпека»

В.о. зав. кафедрою КІСМТ
Марина ЄСІНА
"Допущено до захисту"
« »_____2025 р.

Пояснювальна записка

до кваліфікаційної роботи бакалавра

на тему: «Оцінка системи безпеки WEB-ресурсу QRNG шляхом моделювання реальних атак»

оцінка « _____ »

Голова ЕК

Мичуда Л. З.

Керівник: професор Нарєжній О. П.

Рецензент: старший викладач кафедри комп'ютерних систем та робототехніки Осипчук А. В.

Виконавець: студент групи КБ-42

Близнюк Д. В.

Харків – 2025

РЕФЕРАТ

Пояснювальна записка до бакалаврської кваліфікаційної роботи налічує 60 сторінок, містить 23 рисунки, 4 додатки, а також 41 джерело інформації.

Метою роботи є дослідження рівня безпеки веб-ресурсу QRNG шляхом моделювання дій потенційного зловмисника та реалізація автоматизованого фреймворку для виявлення вразливостей у веб-додатках із використанням сучасних методів та інструментів тестування на проникнення.

Об’єкт дослідження — структура та функціонування веб-ресурсу з погляду кібербезпеки.

Предмет дослідження — підходи до побудови моделі зловмисника, методи аналізу захищеності веб-систем, програмні засоби для проведення пентесту та модульні архітектури автоматизації процесів кібероцінювання.

У дослідженні систематизовано методології веб-тестування безпеки; проаналізовано класифікації вразливостей CWE та методику їх оцінки за шкалою CVSS. Розроблено фреймворк, що автоматизує розвідку, сканування, валідацію та формування звіту. Сканування ресурсу QRNG дозволило виявити типові помилки конфігурації та відсутність сучасних механізмів захисту.

Під час виконання роботи застосовувались методи моделювання загроз, симуляція атаки за сценарієм «чорного ящика», побудова пайплайну на Python, ручне тестування за допомогою Burp Suite, аналіз логів та генерація звітності. Реалізована система дозволяє масштабувати процес перевірки та інтегруватися в CI/CD-процеси.

Результати дослідження можуть використовуватись для періодичного контролю безпеки, в навчальному процесі, а також у практиці аудитів веб-додатків. Запропонований підхід забезпечує гнучкість, відтворюваність і точність під час оцінки захищеності сучасних веб-систем.

Ключові слова: ВЕБ-ПЕНТЕСТ, АВТОМАТИЗАЦІЯ, ВРАЗЛИВОСТІ, CVSS, CWE, OWASP, MITRE, ЗЛОВМИСНИК, ФРЕЙМВОРК, PYTHON, BURP SUITE, ТЕСТУВАННЯ БЕЗПЕКИ.

ABSTRACT

The explanatory note to the bachelor's qualification work consists of 60 pages, contains 23 figures, 4 appendices, and 41 sources of information.

The purpose of the work is to study the security level of the QRNG web resource by modeling the actions of a potential attacker and implementing an automated framework for detecting vulnerabilities in web applications using modern penetration testing methods and tools.

The object of the study is the structure and functioning of the web resource from the point of view of cybersecurity.

The subject of the study is approaches to building an attacker model, methods for analyzing the security of web systems, software tools for conducting pentests, and modular architectures for automating cyber assessment processes.

The study systematizes web security testing methodologies; analyzes the classifications of CWE vulnerabilities and the methodology for assessing them using the CVSS scale. A framework has been developed that automates reconnaissance, scanning, validation, and report generation. Scanning the QRNG resource allowed us to identify typical configuration errors and the lack of modern protection mechanisms.

During the work, threat modeling methods were used, simulating a black box attack scenario, building a pipeline in Python, manual testing using Burp Suite, log analysis, and reporting generation. The implemented system allows you to scale the verification process and integrate into CI/CD processes.

The results of the study can be used for periodic security monitoring, in the educational process, as well as in the practice of web application audits. The proposed approach provides flexibility, reproducibility, and accuracy when assessing the security of modern web systems.

Keywords: WEB PENTEST, AUTOMATION, VULNERABILITY, CVSS, CWE, OWASP, MITRE, ANNULLER, FRAMEWORK, PYTHON, BURP SUITE, SECURITY TESTING.

ЗМІСТ

ПЕРЕЛІК ПОЗНАЧЕНЬ ТА СКОРОЧЕНЬ

OWASP	- Open Web Application Security Project
WSTG	- Web Security Testing Guide
OSSTMM	- Open Source Security Testing Methodology Manual
QRNG	- Quantum Random Number Generator
HTTPX	- HTTP Prober and Information Extractor
CWE	- Common Weakness Enumeration
CVSS	- Common Vulnerability Scoring System
XSS	- Cross-Site Scripting
GDPR	- General Data Protection Regulation
API	- Application Programming Interface
DOM	- Document Object Model
NIST	- National Institute of Standards and Technology
CIA	- Confidentiality, Integrity, Availability (Security Triad)
RFC	- Request for Comments
SQL	- Structured Query Language
SSL	- Secure Sockets Layer
TLS	- Transport Layer Security
HTML	- HyperText Markup Language

ВСТУП

У сучасному цифровому світі, де інформаційні технології проникають у всі сфери життя, питання безпеки веб-додатків постає як одне з ключових. Веб-ресурси використовуються не лише для комунікації, але й для зберігання персональних даних, обробки фінансової інформації, надання критичних державних чи корпоративних послуг. У зв’язку з цим зростає кількість атак, спрямованих саме на веб-рівень — зловмисники активно експлуатують вразливості в логіці додатків, налаштуваннях серверів, а також у неврахованих точках взаємодії між модулями. Це підвищує ризики витоку даних, несанкціонованого доступу, втрати контролю над ресурсами. Зважаючи на постійно зростаючу складність інфраструктури веб-додатків і їх критичну роль, забезпечення їх надійного захисту є одним із головних завдань у сфері кібербезпеки.

Водночас, традиційні заходи захисту, такі як антивіруси чи міжмережеві екрани, вже не здатні повністю запобігти складним і цілеспрямованим атакам. У цьому контексті тестування на проникнення (penetration testing, або пентест) виступає ефективним підходом до оцінки рівня захищеності. Це не лише технічна перевірка — це ціла методологія, яка моделює поведінку реального зловмисника, дозволяючи виявити потенційні слабкі місця ще до того, як ними скористається атакувальний об’єкт. Особливої актуальності пентест набуває у випадках, коли мова йде про відкриті, доступні ззовні системи — саме такими є більшість сучасних веб-ресурсів.

Враховуючи вищезазначене, актуальність теми цієї роботи полягає у необхідності розробки практичного, надійного та масштабованого підходу до тестування безпеки веб-додатків, який поєднує автоматизацію, моделювання реального супротивника та відповідність міжнародним стандартам. У фокусі — розробка фреймворку, що відображає сучасну практику роботи «червоних команд» (red teaming), зокрема, проведення глибокої розвідки, виявлення поверхні атаки, ідентифікації уразливих точок та документування результатів з використанням формалізованих методик, таких як OWASP WSTG, OSSTMM, NIST SP 800-115.

Метою даного дослідження є оцінка рівня безпеки веб-ресурсу QRNG шляхом моделювання реальних атак із використанням сучасних інструментів і методик тестування на проникнення, а також аналіз виявлених вразливостей, їх класифікація та формування рекомендацій щодо усунення. Для реалізації цієї мети передбачалося вирішити такі завдання: здійснити огляд актуальних підходів до тестування веб-додатків; побудувати реалістичну модель зловмисника; створити модульний фреймворк для автоматизованої розвідки та сканування; підібрати, налаштувати та інтегрувати відповідні інструменти (Subfinder, HTTPX, Naabu, Dirsearch, Nuclei, Acunetix); провести практичне тестування на реальному веб-ресурсі; виконати класифікацію виявлених вразливостей за CWE та оцінити їх критичність за CVSS; а також розробити практичні рекомендації з покращення захисту.

Ця робота є прикладом поєднання теоретичних знань і практичного досвіду. Вона демонструє, як за допомогою правильно спроєктованого підходу, структурованої моделі зловмисника та відповідного інструментарію можна здійснити глибоку, послідовну й ефективну перевірку безпеки веб-додатку. Представлений фреймворк, окрім практичного застосування, також може стати корисним прикладом у навчальному процесі або служити основою для подальшого розвитку автоматизованих систем оцінки вразливостей у середовищі DevSecOps.

1 АНАЛІЗ, ОГЛЯД І ДОСВІД ПРОВЕДЕННЯ ПЕРЕВІРОК ВРАЗЛИВОСТЕЙ ТА ІНСТРУЕНТІВ ДЛЯ ТЕСТУВАННЯ БЕЗПЕКИ ВЕБ-РЕСУРСІВ

1.1 Визначення тестування на проникнення

Тестування на проникнення Веб-додатків це структурована безпекова оцінка націлена на симуляцію реальних атак на веб-додаток для ідентифікування та експлуатування вразливостей перш ніж ними зможуть скористатись зловмисники. Головна ціль проведення пентесту – це оцінити стан безпеки додатків шляхом активних спроб обійти захист і отримати несанкціонований доступ до даних або адміністративного функціоналу [1].

На відміну від автоматизованого сканування вразливостей, тестування на веб-проникнення поєднує в собі як і автоматизовані інструменти, так і ручні методи для виявлення складних вразливостей, включаюи вразливості бізнес-логіки, ланцюгові експлойти та пошук помилок в конфігурації, які сканери часто не помічають.

Згідно з OWASP Web Security Testing Guide (WSTG), веб-пентест включає в себе комплексний аналіз таких областей, як автентифікація, управління сеансами, контроль доступу, перевірка вхідних даних, обрброка помилок та конфігурація. Кожна категорія включає чітко визначені тестові кейси, призначені для виявлення слабких місць описаних в OWASP TOP 10 [9].

У своїй книзі «Web Hacking Arsenal», Rafay Baloch описує веб-пентест як дисципліну, що розвивається, яка повинна враховувати користувацьку логіку, хмарні інтеграції та складні взаємодії між інтерфейсом і бекендом. Автор підкреслює, що сучасні веб-додатки – особливо ті, що базуються на API, мікросервісах і фреймворках Javascript – вимагають адаптивних і творчих підходів до тестування [3].

Стандарт NIST SP 800-115 описує пентест як формальний метод, що включає такі етапи, як планування, виявлення, проведення атаки та аналіз після атаки. Такий процес дає певного роду гарантію на те, що вразливості не тільки виявляються, але й перевіряються та визначаються пріоритети на осново реального впливу [10].

OSSTMM (Open Source Security Testing Methodology Manual) ще більше розширює поняття «пентест», вводячи метрики довіри та наголошуючи на точності, правилах застосування та ретельному документуванні. Документ розглядає «пентест» як кількісний процес вимірювання операційної безпеки через об’єктиву взаємодії та аналіз того, як система поводиться під впливом зовнішніх чи внутрішніх дій [6].

1.1.1 Цілі та важливість веб-пентесту

Відповідно до стандарту NIST SP 800-115 пентест забезпечує емпіричнй метод оцінки засобів контролю безпеки шляхом активного тестування компонентів системи з використанням різних методів атак. Це особливо цінне для виявлення не лише технічних недоліків, але й проблем, пов’язаних с процесами, таких як неправильні конфігурації, небезпечні практики розробоки та неадекватні політики контролю доступу [10].

Основними цілями веб-пентесту є:

- Виявлення вразливостей безпеки в коді, конфігурації та логіці програми.
- Оцінка реального впливу потенційної експлуатації шляхом перевірки серйозності та можливості використання кожної вразливості.
- Перевірка ефективності засобів контролю безпеки, включаючи брандмауери, механізми автентифікації та процедури перевірки вхідних даних.
- Демонстрація потенційних ланцюжків атак, в яких кілька проблем меншої серйозності можуть бути об'єднані для досягнення значної компрометації.
- Надання розробникам і власникам систем дієвого зворотного зв'язку та рекомендацій щодо усунення вразливостей.

Як зазначено в The Web Application Hacker's Handbook, тестування на проникнення - це не просто технічний аудит, а засіб розуміння того, як додаток може вийти з ладу в несприятливих умовах. Воно фокусується як на технічних недоліках (таких як SQL-ін'єкція або XSS), так і на логічних вразливостях, таких як порушення бізнес-процесів [5].

З точки зору бізнесу, веб-пентест виконує кілька важливих функцій:

- Зниження ризиків шляхом раннього виявлення та усунення критичних недоліків безпеки.
- Відповідність нормативним вимогам, підтримка таких стандартів, як PCI DSS, ISO/IEC 27001 та GDPR.
- Завоювання довіри клієнтів, особливо для платформ, які обробляють конфіденційні або персональні дані.
- Підготовка до публічних тестувань, таких як програми винагороди за виправлення помилок, шляхом зменшення вразливості.

Зрештою, пентест є наріжним каменем проактивної стратегії кібербезпеки. Це дозволяє організаціям вийти за рамки реактивних заходів безпеки і прийняти мислення, орієнтоване на попередження, виявляючи потенційні точки входу до того, як вони можуть бути використані.

1.1.2 Історична еволюція явища «пентест» та реальні атаки

Еволюція тестування на проникнення до веб-додатків відображає більш широку трансформацію веб-технологій та зростаючу витонченість кіберзагроз. Спочатку веб-додатки були статичними і простими, але в міру того, як наприкінці 1990-х і на початку 2000-х років Інтернет став більш інтерактивним і орієнтованим на комерцію, зловмисники почали використовувати нові динамічні функції.

Ранні атаки часто були спрямовані на погано перевірені вхідні дані через SQL-ін'єкції, міжсайтовий скриптинг (XSS) та «directory traversal». Ці типи вразливостей виникли через брак безпечних методів розробки та недостатню обізнаність про загрози на веб-рівні. У відповідь на це, такі ініціативи, як OWASP Top 10 (вперше випущений у 2003 році), почали формалізувати та класифікувати загальні проблеми безпеки, що поклало початок усвідомленню спільнотою безпеки додатків.

З появою AJAX, Web 2.0 та односторінкових додатків (SPA) додатки ставали дедалі складнішими, а зловмисники адаптували свої методи. У книзі «Web Hacking Arsenal», Rafay Baloch розповідає про те, як ці архітектурні зміни призвели до появи нових векторів атак, таких як XSS на основі DOM, маніпуляції з токенами та зловживання бізнес-логікою. Тестувальники проникнення почали зосереджуватися

не лише на технічних недоліках, але й на вразливостях у тому, як додаток логічно обробляє робочі процеси та вхідні дані користувачів.

Низка значущих атак, наведена у таблиці В.1 у Додатку В, у реальному середовищі відображає хід цієї еволюції:

Випадки наведені у таблиці В.1 Додатку В підкреслюють важливу тенденцію: сучасні веб-атаки часто поєднують кілька вразливостей - наприклад, ланцюговий обхід авторизації з фіксацією сеансу або використання JavaScript-ін'єкцій для перехоплення потоків OAuth. Крім того, сьогодні зловмисники активно атакують API, мікросервіси та хмарні розгортання, які є складними і часто погано контрольованими об'єктами атак.

У фреймворку OSSTMM зазначається, що розуміння минулих атак має вирішальне значення для моделювання загроз і розробки ефективних процедур тестування. Глибоке розуміння історичних методів допомагає тестувальникам передбачити, як зловмисники можуть використовувати нові технології або неправильні конфігурації.

Таким чином, історія атак на веб-додатки - це історія адаптації як з боку зловмисників, так і з боку захисників. Тестування на проникнення продовжує розвиватися як проактивний механізм захисту, допомагаючи організаціям зрозуміти свої слабкі місця до того, як це зроблять зловмисники.

1.2 Види та моделі пентесту

Методології тестування на проникнення можуть суттєво відрізнятися залежно від цілей, обсягу та рівня доступу, наданого тестувальнику. Розуміння різних моделей і типів тестів на проникнення має вирішальне значення для вибору відповідної стратегії оцінки стану безпеки веб-додатків. Ці моделі відрізняються в першу чергу кількістю інформації, що надається тестувальнику, і перспективою, з якої аналізується система.

1.2.1 Тестування «чорного ящика», «білого ящика», «сірої скриньки»

Тестування «чорного ящика» імітує дії неавторизованого зовнішнього зловмисника, який не має попередніх знань про внутрішню структуру, вихідний код або конфігурацію системи. Тестувальник взаємодіє з додатком ззовні,

використовуючи загальнодоступну інформацію та методи розвідки для виявлення вразливостей.

Цей підхід наближений до реальних сценаріїв атак і допомагає оцінити безпеку периметру додатку.

Як зазначається в Арсеналі веб-хакінгу, тестування чорного ящика ідеально підходить для тестування реального виробничого середовища та оцінки того, що може побачити типовий зловмисник. Однак воно має обмежену видимість на рівні коду, що може призвести до меншого покриття вразливостей, особливо для глибоких логічних помилок або вразливостей контролю доступу, які лежать за рівнями автентифікації.

Тестування «білого ящика», яке також називають тестуванням повних знань, дає тестувальнику повний доступ до внутрішніх ресурсів, таких як:

- Вихідний код та конфігураційні файли
- Архітектурна документація
- Облікові дані адміністратора та ключі API

Ця модель використовується для проведення ретельної перевірки безпеки та виявлення складних вразливостей, які можуть бути невидимими ззовні. Вона дозволяє проаналізувати: небезпечні практики кодування; приховані бекдори; недосконалу логіка в бізнес-процесах; припущення щодо довіри до API.

Згідно з OSSTMM, тестування за допомогою білих скриньок підвищує точність і відтворюваність тестування [6].

Тестування «сірої скриньки» пропонує золоту середину, в якій тестувальник має часткові знання про систему - наприклад, облікові дані користувача або документацію API. Він імітує зловмисника з обмеженим доступом, а саме: автентифікаваного користувача, стороннього інтегратора або інсайдера з обмеженими привілеями. Ця модель особливо ефективна для тестування – системи управління доступом на основі ролей (RBAC - Role-based access control).

Тестування за допомогою сірої скриньки балансує між реалістичністю і глибиною, що робить його популярним в корпоративному середовищі, де

тестувальники повинні оцінити як зовнішні ризики, так і внутрішні межі довіри без повного доступу до вихідного коду.

1.2.2 Bug Bounty платформи проти формального пентесту

Оскільки організації прагнуть проактивно виявляти та усувати вразливості в системі безпеки, з'явилися два взаємодоповнюючі підходи: формальний тест на проникнення та платформи bugbounty. Хоча обидва підходи спрямовані на виявлення вразливостей до того, як ними скористаються зловмисники, вони суттєво відрізняються за методологією, структурою та операційним контекстом.

Формальне тестування на проникнення — це чітко регламентована оцінка безпеки, що виконується внутрішніми або зовнішніми фахівцями за заздалегідь визначеним обсягом, тривалістю та методикою. Таке тестування зазвичай узгоджується зі стандартами, як-от NIST SP 800-115, OSSTMM або OWASP WSTG. Воно проводиться у контрольованому середовищі з використанням підготовлених облікових записів і охоплює перевірку конкретних цілей у визначених часових межах.

Формальний пентест часто застосовується для дотримання вимог стандартів, перед запуском нових застосунків, при оцінці критичних систем або після інцидентів для перевірки усунення вразливостей. Його результати зазвичай добре документовані, відтворювані й містять аналіз ризиків із рекомендаціями щодо усунення. Це робить тестування корисним як для технічних фахівців, так і для управлінського рівня.

Програми винагород за виявлені вразливості (bug bounty) залучають зовнішніх дослідників з усього світу до тестування безпеки реальних застосунків в обмін на фінансову винагороду. Такі платформи, як HackerOne, Bugcrowd чи YesWeHack, забезпечують технічну й юридичну основу для прийому, перевірки та опрацювання звітів [7].

На відміну від формальних тестів, баг-баунті базуються на принципі відкритого конкурсу: чим цікавіша вразливість — тим вища винагорода. Це створює умови для постійного, творчого тестування, яке іноді дозволяє виявити нестандартні або складні проблеми. Водночас, відсутність контролю за глибиною

аналізу, ризики дублювання, слабка модерація якості звітів і потреба у внутрішньому сортуванні результатів можуть стати проблемами для організацій [7].

Такі програми найкраще підходять для систем, що вже працюють у продакшені, мають значну поверхню атаки, здатні оперативно реагувати на повідомлення про вразливості.

Порівняльні характеристики обох підходів наведені у таблиці В.2 у Додатку В.

На практиці багато організацій застосовують гібридний підхід, використовуючи формальні тести на проникнення для структурованих оцінок і баг-баунті програми для безперервного тестування на вразливість в реальних умовах. При правильному поєднанні обидві моделі роблять значний внесок у створення надійної та адаптивної системи безпеки.

1.2.3 Внутрішнє та зовнішнє оцінювання

При тестуванні на проникнення перспектива атаки відіграє вирішальну роль у визначенні обсягу та глибини оцінки. Для моделювання різних позицій зловмисника і моделей загроз зазвичай використовуються два основних типи оцінок - зовнішня і внутрішня - для моделювання різних позицій зловмисника і моделей загроз.

Зовнішнє оцінювання в тестуванні на проникнення передбачає моделювання атаки з боку зловмисника, який діє повністю за межами внутрішньої мережі організації. Такий підхід імітує дії кіберзлочинців або хактивістів, які шукають публічно доступні вразливості в інтернеті. Оцінка проводиться без внутрішнього доступу, переважно зосереджуючись на зовнішніх сервісах — веб-додатках, API, VPN-порталах, DNS, поштових серверах тощо.

Процес включає збір відкритої інформації, сканування портів, аналіз конфігурацій SSL/TLS, грубий перебір облікових даних і тестування виявлених вразливостей. Основна мета — з’ясувати, наскільки ефективно організація захищена від зовнішніх атак, та чи функціонують системи периметрового захисту (WAF, міжмережеві екрани, DDoS-захист).

Відповідно до рекомендацій NIST SP 800-115, саме такі перевірки мають першочергове значення для структур, що надають критичні онлайн-сервіси, оскільки дають змогу оцінити їхню стійкість до несанкціонованого доступу ззовні [10].

Тест на внутрішнє проникнення імітує зловмисника, який вже прорвався через периметр - за допомогою успішного фішингу, скомпрометованих облікових даних або фізичного доступу - або представляє зловмисного інсайдера (наприклад, незадоволеного співробітника або підрядника).

Головні характеристики:

- Виконується з внутрішньої мережі (LAN/VPN)
- Націлені на інтранет-портали, внутрішні API, файлові ресурси, Active Directory, внутрішні бази даних, пристрої IoT та кінцеві системи
- Оцінює такі ризики, як ескалація привілеїв, бічне переміщення(lateral movement), несанкціонований доступ до ресурсів і збої сегментації.

Порівняльні характеристики обох підходів наведені у таблиці В.3 у Додатку В.

Разом внутрішні та зовнішні тести на проникнення дають комплексне уявлення про стан безпеки організації, охоплюючи як захист кордонів, так і внутрішній контроль.

1.3 Міжнародні стандарти та методології тестування

З розвитком тестування на проникнення веб-додатків було розроблено кілька міжнародних стандартів і фреймворків, які допомагають фахівцям з безпеки планувати, виконувати і документувати тести на проникнення. Ці методології пропонують структуровані підходи, забезпечують узгодженість і допомагають узгодити зусилля з тестування з регуляторними та організаційними цілями безпеки.

У цьому розділі розглядаються чотири основні методології (рис. 1.1), які зазвичай використовуються для оцінювання веб-безпеки: OSSTMM, ISSAF, OWASP WSTG та NIST SP 800-115. Кожна з них пропонує унікальну перспективу і структуру, що відповідає різним цілям тестування і потребам організації.

Рисунок 1.1 - Характеристика методологій тестування на проникнення

Посібник OSSTMM, розроблений ISECOM, є одним із визнаних стандартів тестування безпеки, який акцентує увагу на вимірюваності, повторюваності та об'єктивності результатів. Його особливістю є впровадження метрик довіри — кількісної оцінки взаємодії між об’єктами в системі безпеки. Завдяки таким характеристикам, як симетрія, узгодженість і видимість, організації можуть визначити зони «сліпої довіри» та посилити захист на основі обґрунтованого аналізу [6].

Окрім технічної складової, OSSTMM передбачає чіткі правила взаємодії, що регулюють етичні, юридичні та процедурні аспекти тестування. Це, зокрема, вимога письмового дозволу на проведення тестів, дотримання конфіденційності, уникнення агресивного маркетингу й чітке визначення зони відповідальності в контрактах. Усе це формує прозорий і контрольований підхід до оцінки безпеки, що відповідає сучасним вимогам до професійного пентесту [6].

Методологія ISSAF, створена OISSG, пропонує структурований і модульний підхід до оцінки безпеки, охоплюючи як технічні, так і організаційні аспекти. Вона складається з трьох основних етапів: планування, активного тестування та звітування з очищенням.

Під час планування визначається обсяг, узгоджуються умови тестування та оформлюються юридичні дозволи. На етапі тестування реалізується багатоетапний цикл, що охоплює збір інформації, виявлення вразливостей, проникнення, ескалацію привілеїв, підтримку доступу та маскування слідів — з урахуванням

поведінки реального зловмисника. Завершальний етап включає складання звіту, повідомлення про критичні проблеми й очищення середовища від артефактів [17].

Оцінювання в ISSAF проводиться за модулями, що адаптуються до конкретних цілей — від веб-додатків до фізичної безпеки чи соціальної інженерії. Незважаючи на те, що методологія поступається популярністю OWASP чи NIST, вона залишається ефективною для глибоких аудитів і корпоративного аналізу ризиків [17].

OWASP Web Security Testing Guide (WSTG) — це спеціалізований стандарт, орієнтований на тестування безпеки саме веб-додатків. Версія 4.2, випущена у 2020 році, стала основним орієнтиром для ручного пентестингу в сучасному веб-середовищі, особливо там, де використовуються API, JavaScript-фреймворки та динамічні інтерфейси [9].

На відміну від універсальних методологій, як-от NIST чи OSSTMM, WSTG повністю фокусується на веб-рівні. Його структура включає 14 категорій тестів, кожна з яких охоплює окремий аспект безпеки — від збору інформації до тестування бізнес-логіки та обробки помилок. Кожен тест містить чітко описані цілі, приклади корисного навантаження, індикатори вразливостей і прив’язки до OWASP Top 10 або CVE [9].

WSTG добре підходить для ручного тестування, API-аналізу, інтеграції в CI/CD-конвеєри та валідації безпечного життєвого циклу розробки. Його практичний формат і деталізація роблять його надійною основою для систематичного та повторюваного тестування веб-додатків [9].

Методологія NIST SP 800-115 пропонує чіткий, поетапний підхід до тестування безпеки, що охоплює всі ключові аспекти пентесту. Процес починається з етапу планування, під час якого визначаються цілі, правила взаємодії, обсяг робіт і комунікаційні протоколи. Це забезпечує узгодженість дій усіх учасників до початку будь-яких технічних дій.

Далі виконується етап виявлення, який охоплює збір інформації про систему та аналіз вразливостей — як автоматизованими, так і ручними засобами. Після

цього настає фаза атаки, де тестувальники перевіряють знайдені вразливості на практиці, оцінюють потенційний вплив і можливість подальшого проникнення.

Завершальним елементом є звітування, яке триває протягом усього процесу й завершується складанням повного технічного та управлінського звіту з рекомендаціями для пом’якшення ризиків. Така структура дозволяє проводити тестування системно, відтворювано і з урахуванням як технічних, так і бізнес-аспектів.

1.4 Ключові теоретичні поняття

Надійний процес тестування на проникнення до веб-сторінок ґрунтується не лише на практичних інструментах і технічних робочих процесах, але й на добре відпрацьованих теоретичних засадах. Ці концепції забезпечують структуровану лінзу, за допомогою якої можна ідентифікувати, класифікувати, аналізувати та розставляти пріоритети. Без міцного теоретичного підґрунтя результати оцінки безпеки можуть бути непослідовними, неповними або неправильно інтерпретованими.

Теоретичні моделі, відіграють вирішальну роль у формуванні того, як моделюються загрози, як оцінюються вразливості з точки зору впливу і серйозності, і як в кінцевому підсумку оцінюються ризики і повідомляються зацікавленим сторонам. Вони не є абстрактними академічними конструкціями - скоріше, вони є невід'ємною частиною реальної практики безпеки, вбудованою в методології, такі як розроблені NIST, OWASP та іншими визнаними організаціями.

Основні концептуальні області, що мають відношення до тестування на проникнення, включають в себе наступні:

- Моделювання загроз, яке використовується для передбачення та класифікації потенційної поведінки та цілей зловмисників. Сюди входять структуровані підходи, такі як STRIDE і DREAD, які підтримують систематичну ідентифікацію та аналіз ризиків різних типів загроз на основі технічних і бізнес-критеріїв.

- Тріада (CIA) та CVSS, які пропонують основоположні принципи для розуміння наслідків недоліків безпеки та присвоєння рейтингів серйозності, які визначають пріоритети виправлення.
- Методології оцінки ризиків, зокрема від NIST та OWASP, які забезпечують узгоджені рамки для аналізу ризиків для безпеки з точки зору ймовірності, впливу та контексту.

Разом ці моделі дозволяють тестувальникам на проникнення підходити до своєї роботи з аналітичною строгістю і надавати значущі, дієві висновки на основі виявлених ними вразливостей.

1.5 Звітність про випробування та відповідальне розкриття інформації

Заключний етап тесту на проникнення передбачає не лише документування результатів, але й забезпечення інформування про виявлені вразливості та управління ними таким чином, щоб сприяти своєчасному усуненню та мінімізації ризиків. Добре підготовлений звіт перетворює необроблені результати на дієву інформацію, а відповідальна практика розкриття інформації допомагає гарантувати, що проблеми безпеки вирішуються без зайвої шкоди для організації, що постраждала.

Ефективні звіти про тестування слугують кільком цілям: вони містять технічний запис процесу тестування, детальний опис та оцінку впливу кожної вразливості, а також пріоритетні рекомендації щодо її усунення. Звіти повинні бути адаптовані до своєї аудиторії - технічні команди потребують детальних підтверджень концепції, в той час як зацікавлені сторони на рівні керівництва отримують вигоду від узагальнень, які фокусуються на впливі на бізнес і рівнях ризиків.

Сучасна звітність часто базується на встановлених стандартах і рекомендаціях, таких як NIST SP 800-115 і OSSTMM (Open Source Security Testing Methodology Manual). Ці стандарти допомагають визначити, яку інформацію слід включати до звіту з оцінки безпеки, як його структурувати і як забезпечити чіткість і послідовність.

Окрім документування, процес відповідального розкриття вразливостей відіграє важливу роль у підтримці етичних та професійних стандартів у сфері кібербезпеки. Відповідальне розкриття - це скоординований процес повідомлення про недоліки безпеки постраждалим сторонам, що дає їм час на впровадження виправлень до того, як відбудеться публічне розкриття або використання. Він регулюється офіційними протоколами, такими як RFC 9116 і RFC 2350 і підтримується такими платформами, як Bugcrowd і HackerOne [12-13].

У цьому розділі описано ключові елементи професійного звітування про результати тестів на проникнення та принципи етичного розкриття вразливостей. Він охоплює структуру та компоненти звіту, використання оцінок CVSS та візуальних доказів для підтвердження висновків, а також механізми, за допомогою яких можна відповідально повідомляти про вразливості.

1.5.1 Структура звіту від NIST та OSSTMM

Підхід NIST передбачає чіткий поділ на розділи, які охоплюють як технічну, так і управлінську складову: резюме для керівництва, опис обсягу тестування, методики, результати з доказами (PoC, CVSS), оцінку впливу на бізнес, рекомендації та додатки з доказами. Такий формат дозволяє зручно комунікувати як з технічними фахівцями, так і з менеджментом.

На відміну від цього, OSSTMM орієнтується на кількісну оцінку. Звіти за цією методикою зосереджуються на показниках довіри, метриках ефективності контролю та повторюваності тестів. Це дає змогу незалежно перевіряти результати та будувати більш формалізовані висновки щодо операційної безпеки.

Обидва підходи мають свої переваги: NIST — більш наочний і прикладний, OSSTMM — строгий і метрикорієнтований. Добре структурований звіт - чи то описовий (NIST), чи то на основі метрик (OSSTMM) - відіграє життєво важливу роль у наданні організаціям дієвих результатів.

1.5.2 Proof of Concept (PoC)

Якість звіту про тестування на проникнення значною мірою визначається наявністю чітких і переконливих доказів, що підтверджують виявлені вразливості. Просто зазначити факт її існування недостатньо — важливо продемонструвати, як

саме вразливість була виявлена, чи можна її використати на практиці, і які наслідки це може мати.

Цю роль виконують приклади Proof of Concept (PoC) — відтворювані сценарії, які ілюструють експлуатацію уразливості. До них зазвичай входять URL-адреси, запити, скрипти та детальний опис кроків відтворення. Це дозволяє командам безпеки або розробникам перевірити проблему самостійно й оцінити її серйозність.

Не менш важливими є скріншоти, які забезпечують наочність. Вони особливо корисні у випадках, коли йдеться про користувацький інтерфейс або витік даних. Найкраща практика — додавати до них контекст, часові позначки та підписи без зайвого редагування.

Таким чином, поєднання PoC та візуальних доказів не лише робить звіт професійним і зрозумілим, а й підвищує його практичну цінність. Це відповідає рекомендаціям NIST SP 800-115 і OSSTMM, які акцентують увагу на повторюваності та перевірюваності результатів.

1.5.3 Оцінка CVSS та вплив на бізнес

Оцінка серйозності вразливостей є важливою складовою професійного тестування на проникнення. Для цього використовується система CVSS (Common Vulnerability Scoring System), яка надає числову оцінку загрозі (від 0.0 до 10.0), враховуючи технічні параметри, складність експлуатації, потребу в привілеях, взаємодію з користувачем та вплив на конфіденційність, цілісність і доступність.

Для прикладу: критична вразливість SQL-ін’єкції на публічній формі входу без автентифікації може отримати базовий бал 9.8 — її можна використати віддалено, без привілеїв, і вона впливає на всі три компоненти безпеки.

Однак навіть така висока технічна оцінка не відображає повної картини без врахування бізнес-контексту. Якщо мова йде про онлайн-банкінг або обробку персональних даних, ризик зростає ще більше через можливі наслідки — фінансові втрати, штрафи за невідповідність (наприклад, GDPR), чи репутаційні збитки.

Тому ефективна звітність повинна поєднувати обидва рівні — технічний (CVSS) і стратегічний (бізнес-аналіз). Такий підхід забезпечує баланс між

технічною точністю і практичною цінністю для прийняття рішень. Він відповідає найкращим практикам, викладеним у NIST SP 800-115, OWASP та інших сучасних фреймворках [40].

1.5.4 Порівняння підходів звітності

Оцінка вразливостей у звіті про пентест не обмежується лише їх виявленням — вона включає кілька важливих компонентів, які разом формують повноцінну аналітичну картину. Зокрема, CVSS надає технічну оцінку серйозності вразливості, Proof of Concept (PoC) демонструє її реальну експлуатацію, а структура звіту, відповідно до стандартів NIST та OSSTMM, забезпечує системність подачі й можливість відтворення результатів. Порівняння ключових характеристик цих елементів наведено у таблиці В.4 у Додатку В.

1.6 Висновки за розділом

У результаті аналізу першого розділу було систематизовано сучасні підходи до тестування на проникнення веб-додатків, розглянуто класифікацію методів, моделі тестування та їх особливості. Встановлено, що веб-пентест є критично важливою складовою сучасної стратегії кібербезпеки, оскільки дозволяє не лише виявити технічні вразливості, але й оцінити їхній вплив у реальному бізнес-контексті.

Проведено порівняльний аналіз формальних пентестів та баг-баунті програм, що дозволило виявити переваги та обмеження кожного підходу залежно від цілей і ресурсів організації. Окремо досліджено зовнішнє та внутрішнє оцінювання як ключові вектори моделювання атак з точки зору зловмисника.

Особливу увагу приділено міжнародним методологіям, зокрема OSSTMM, ISSAF, OWASP WSTG та NIST SP 800-115. Було встановлено, що кожна з них має власну специфіку: від формалізованої звітності й метрик (OSSTMM) до практично орієнтованих покрокових підходів (OWASP WSTG, NIST). Також розглянуто теоретичні основи моделювання загроз, оцінки ризиків та формування звітної документації з урахуванням стандартів CVSS та PoC-структури.

2 РОЗРОБКА МОДЕЛІ ПОРУШНИКА ДЛЯ АТАК НА ВЕБ-РЕСУРСИ ТА РОЗРОБКА ФРЕЙМВОРКА ДЛЯ АВТОМАТИЗОВАНОГО ЗБОРУ ІНФОРМАЦІЇ

2.1 Моделювання загроз та зловмисників

Тестування на проникнення до веб-додатків стає по-справжньому ефективним, коли воно керується реалістичною, чітко визначеною моделлю зловмисника та чітким набором цілей. Замість того, щоб розглядати пентестування як суто технічний перелік вразливостей, сучасні підходи наголошують на моделюванні поведінки зловмисників з урахуванням ризиків, що відображає, як реальні зловмисники націлені на систему, які ресурси вони можуть використовувати і які цілі вони переслідують. Для ефективного моделювання процес починається з моделювання загроз.

Моделювання загроз - це систематичний процес, який використовується для виявлення, класифікації та оцінки потенційних загроз для програми або системи. В оцінці кібербезпеки воно слугує структурованим методом передбачення того, як зловмисники можуть взаємодіяти з ціллю, які активи їх цікавлять, і як ці активи можуть бути викриті або використані. Інтегруючи моделювання загроз у тестування на проникнення, тестувальники гарантують, що їхні зусилля є стратегічними, реалістичними та узгоджуються з фактичною поведінкою зловмисників, а не з загальним переліком вразливостей.

У цьому розділі описано підхід до моделювання, використаний у цьому дослідженні, а саме, як розробляється реалістична модель зловмисника для атаки на обраний веб-ресурс https://www.qrngua.website/. Всі засоби розвідки, сканування та автоматизації, розроблені пізніше в цій роботі, визначаються характеристиками та цілями цього змодельованого зловмисника.

Підходячи до тестування з точки зору конкретного зловмисника, ця робота гарантує, що кожна технічна дія - від переліку доменів до сканування вразливостей - узгоджується з послідовною та реалістичною стратегією. Такий підхід також сприяє створенню більш змістовних звітів і дозволяє розставляти пріоритети в

результатах тестування на основі фактичного ризику, а не лише теоретичної серйозності загрози.

2.1.1 Мета моделювання загроз при тестуванні на проникнення

Моделювання загроз є важливим підготовчим етапом у процесі тестування на проникнення. Воно дозволяє фахівцям з безпеки вийти за рамки простого виявлення вразливостей і перейти до структурованого моделювання реалістичної поведінки зловмисника. Визначаючи ймовірних зловмисників, їхні цілі, можливості та потенційні шляхи атаки, моделювання загроз узгоджує діяльність з тестування з реальними, а не теоретичними ризиками.

Основна мета моделювання загроз при тестуванні на проникнення полягає в тому, щоб зосередити зусилля з тестування на областях з найбільшим впливом і ймовірністю, ґрунтуючись на обґрунтованих припущеннях про те, як діятиме реальний супротивник. Це призводить до більш ефективних і значущих результатів, оскільки пріоритетом є не лише технічна складність, але й бізнес-релевантність та можливість використання уразливостей.

Ключові функції та переваги моделювання загроз у цьому контексті включають в себе наступні:

1) Визначення профілів зловмисників

Моделювання загроз дозволяє тестувальникам встановити типи зловмисників з певними характеристиками: наприклад, чи є зловмисник внутрішнім або зовнішнім, аутентифікованим або анонімним, опортуністичним або цілеспрямованим. Ці профілі допомагають імітувати різну поведінку зловмисника при різних рівнях доступу та мотивації.

2) Зосередження на критичних активах та векторах атак

Замість того, щоб намагатися оцінити всю систему однаково, моделювання загроз виділяє конкретні активи і точки входу, які, найімовірніше, будуть атаковані. Це можуть бути вразливі API, механізми автентифікації або області, що обробляють конфіденційні дані. Результатом є більш цілеспрямований підхід до тестування, заснований на оцінці ризиків.

3) Трансформація технічних вразливостей у бізнес-ризики

Оцінки безпеки часто не можуть передати вплив в термінах, зрозумілих для зацікавлених сторін. Моделювання загроз допомагає пов'язати технічні висновки (наприклад, SQL-ін'єкції) з реальними наслідками (наприклад, втрата даних, порушення нормативних вимог, операційні збої), покращуючи як визначення пріоритетів, так і реагування на них.

4) Структурування моделі

Після визначення моделі зловмисника, тестування на проникнення стає структурованою симуляцією діяльності противника. Дії тестувальника - такі як сканування, перерахування та доставка корисного навантаження - базуються на сценаріях загроз, а не на довільному використанні інструментів. Це робить методологію тестування відтворюваною та узгодженою з галузевими стандартами, такими як NIST SP 800-115 та OWASP WSTG.

5) Покращення зрозумілості звіту та комунікації із зацікавленими сторонами

Чітке моделювання загроз підвищує прозорість протягом усього процесу. Це дозволяє пояснити у звіті про тест на проникнення не лише те, які вразливості були знайдені, але й те, чому ці проблеми є важливими в реалістичному контексті загроз. Це сприяє кращому прийняттю рішень та плануванню усунення недоліків.

Таким чином, моделювання загроз забезпечує стратегічний контекст, необхідний для якісного тестування на проникнення. Воно керує моделюванням дій зловмисника, вибором інструментів, інтерпретацією даних і передачею результатів. Це гарантує, що тестування відображає умови реальних атак, а також узгоджується з цілями і обмеженнями організації, що оцінюється.

2.1.2 Моделювання загроз (STRIDE, DREAD)

Моделювання загроз - це проактивний підхід, який використовується в кібербезпеці для систематичного виявлення, класифікації та оцінки потенційних загроз для системи. У контексті тестування на проникнення до веб-додатків воно слугує основою для прогнозування поведінки зловмисників і спрямування оцінки на зони найбільшого ризику. Серед найбільш поширених моделей моделювання загроз - STRIDE і DREAD.MITRE ATT&CK.

Модель STRIDE, розроблена компанією Microsoft, класифікує загрози на шість основних типів (рис. 2.1):

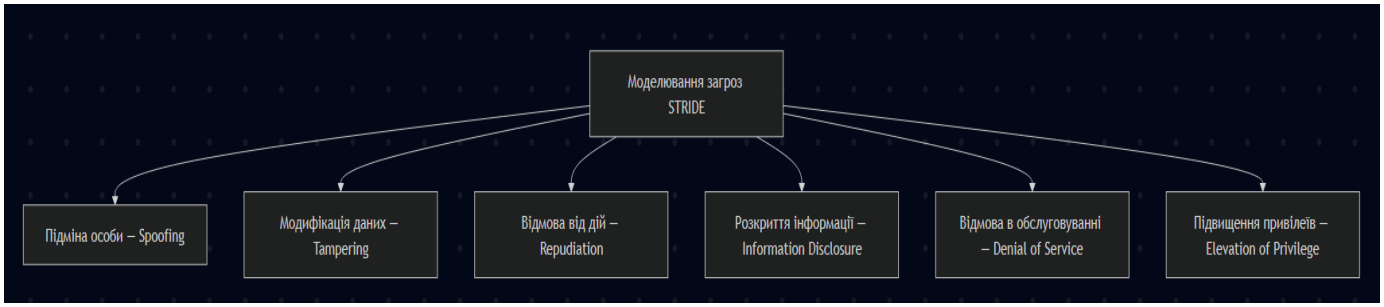


Рисунок 2.1 – Опис моделі загроз STRIDE

- Спуфінг - видавання себе за когось іншого (наприклад, обхід входу в систему)
- Фальсифікація - несанкціонована модифікація даних (наприклад, маніпуляції з параметрами)
- Заперечення - заперечення дій без доказів (наприклад, відсутність логів)
- Розкриття інформації - розголошення конфіденційних даних (наприклад, витік даних)
- Відмова в обслуговуванні(DoS – Denial of Service) - порушення доступності послуг (наприклад, переповнення запитів)
- Підвищення привілеїв - отримання несанкціонованого доступу або вищих привілеїв

STRIDE особливо корисний під час проектування або перегляду системи, щоб гарантувати, що міркування безпеки будуть враховані для різних поверхонь атаки [37].

DREAD - це система оцінки загроз, яка використовується для визначення пріоритетності та кількісної оцінки ризиків на основі п'яти критеріїв (рис. 2.2):

Рисунок 2.2 – Опис моделі загроз DREAD

- Потенційна шкода - Якою буде ступінь шкоди, якщо загроза буде реалізована?
- Відтворюваність - наскільки легко відтворити атаку?
- Експлуатабельність - наскільки легко здійснити атаку?
- Користувачі, на яких це вплине - скільки користувачів постраждає?
- Можливість виявлення - наскільки легко виявити вразливість?

Кожен фактор зазвичай оцінюється від 1 до 10, а середнє значення дає числову оцінку ризику, що допомагає тестувальникам і зацікавленим сторонам визначити пріоритетність вразливостей, які слід усунути в першу чергу.

Разом STRIDE і DREAD дозволяють проводити структурований аналіз загроз і приймати обґрунтовані рішення на основі якісних і кількісних оцінок.[38]

Система MITRE ATT&CK (Adversarial Tactics, Techniques & Common Knowledge) - це база знань про спостережувані тактики і методи, що використовуються суб'єктами загроз. На відміну від STRIDE і DREAD, вона базується на реальних даних про кібератаки.

MITRE ATT&CK широко використовується для моделювання поведінки зловмисників у «red team», емуляції супротивника та розробки сценаріїв цільового тестування на проникнення. Він підтримує узгодження тестових процедур з відомими шляхами атаки, покращуючи релевантність та повноту загроз [41].

На практиці моделі зазначені у таблиці Г.1 у додатку Г можна використовувати в комбінації. Наприклад, STRIDE допомагає визначити, що тестувати, DREAD оцінює, наскільки серйозними є результати, а MITRE ATT&CK гарантує, що тести відображають відому поведінку зловмисників. Їх інтеграція покращує релевантність тестів, ефективність і загальну якість процесу оцінки загроз.

2.1.3 Побудова індивідуальної моделі зловмисника та визначення об'єктів тестування

Реалістичний тест на проникнення вимагає більше, ніж серії автоматизованих сканувань - він вимагає моделювання розумного, цілеспрямованого супротивника, здатного адаптуватися до контексту,

інтерпретувати поведінку програми та використовувати вразливості, де це необхідно. Для цього розроблена структурована модель зловмисника, яка відображає, як реальні суб'єкти загроз підходять до тестування веб-додатків.

Ця модель спирається на усталені методології, такі як Penetration Testing Execution Standard (PTES), MITRE ATT&CK, NIST SP 800-115, OSSTMM, а також на практичні робочі процеси, що використовуються фахівцями з наступальної безпеки. Вони визначають всю структуру системи автоматизації та підходи до тестування, що використовуються в цьому дослідженні.

У нашому випадку модель зловмисника моделюється як зовнішній, неавторизований користувач «чорного ящика», оснащений інструментами з відкритим вихідним кодом, обмеженими ресурсами і без попереднього доступу або облікових даних. Його поведінка структурована на шість тактичних фаз:

Етапи методології зловмисника:

1) Розвідка. Зловмисник збирає загальнодоступну інформацію про цільову інфраструктуру. Сюди входять пасивні дані з DNS, SSL/TLS, WHOIS та відкритих метаданих, а також активне мапування за допомогою таких інструментів, як Subfinder, Waymore та httpx. Мета полягає в тому, щоб перерахувати субдомени, кінцеві точки та технології, що використовуються, формуючи повну карту веб-додатків.
2) Сканування. Такі інструменти, як Naabu, dirsearch та Acunetix, використовуються для виявлення відкритих портів, перерахування каталогів, ідентифікації веб-фреймворків та виявлення прихованих або застарілих кінцевих точок. Цей етап допомагає визначити доступну поверхню атаки, виділяючи компоненти, які мають бути пріоритетними в більш глибокому тестуванні.
3) Аналіз вразливостей. Використовуючи інструменти такі як nuclei так Acunetix сканує:
    - Відомі CVE, пов’язані з ідентифікованим програмним забезпеченням
    - Топ-10 вразливостей OWASP (XSS, SQLi, SSRF та ін.)

- Неправильні конфігурації безпеки, такі як відкриті файли .env або облікові дані за замовчуванням. Цей етап допомагає швидко впорядкувати слабкі місця програми перед будь-якою спробою експлуатації.

4) Експлуатація використовуючи Burp Suite. Після виявлення потенційних вразливостей зловмисник намагається використати їх вручну, щоб підтвердити вплив і можливість реалізації. Хоча автоматизовані інструменти можуть забезпечити базову перевірку, ручне тестування за допомогою Burp Suite є ключовим на цьому етапі.

   Burp Suite дозволяє:

   - Перехоплення та модифікація HTTP-запитів (тестування логіки авторизації/сесії)
   - Маніпулювання вхідними даними та токенами в потоках входу, API та формах
   - Fuzzing параметрів за допомогою Burp Intruder, щоб викликати XSS, SQLi або IDOR
   - Повторення та автоматизація корисного навантаження експлойтів за допомогою Burp Repeater або розширень
   - Дослідження логічних помилок, небезпечних перенаправлень або умовної поведінки
   - Обхід засобів захисту шляхом ручного аналізу JavaScript, токенів або заголовків

Цей практичний контроль забезпечує достовірність, відтворюваність і придатність результатів до використання в реалістичних умовах. Наприклад, якщо nuclei виявляє потенційно вразливу кінцеву точку, Burp Suite використовується для уточнення корисного навантаження і підтвердження експлуатації.

5) Пост-експлуатація. Якщо експлуатація успішна, зловмисник вивчає подальші наслідки::

   - Доступ до внутрішніх кінцевих точок або адміністративних панелей
   - Вплив на чутливі дані

- Місця завантаження файлів або витоку вихідного коду Хоча ця фаза обмежена за обсягом у багатьох операціях, вона все ж моделюється для відображення того, як реальні зловмисники розширюють доступ після отримання початкового плацдарму.

6) Моделювання звітності та розкриття інформації. Нарешті, зловмисник збирає всі результати, корисне навантаження для PoC, скріншоти та логи запитів. Звіти моделюються у структурованих форматах (наприклад, резюме на основі CVSS) і відображають відповідальні практики розкриття інформації. Цей крок гарантує, що результати тесту можна використовувати, відстежувати та визначати пріоритети.

У цьому дослідженні симульованому зловмиснику було поставлено завдання атакувати певний веб-ресурс: www.qrngua.website. Це контрольоване середовище, до якого надано повний доступ з метою оцінки безпеки. Тестування проводиться в межах правових та етичних норм, і жодних спроб несанкціонованого доступу за межами вказаного домену не робиться.

Встановлюючи цілі, модель зловмисника залишається узгодженою як з етичними межами, так і з реалістичною поведінкою атаки. Система автоматизації та ручна перевірка, описана в наступних розділах, призначена для досягнення цих цілей за допомогою структурованого сканування, експлуатації та аналізу.

Діаграма на рисунку 2.3 окреслює тактичний потік дій зловмисника, який рухається від поверхневого виявлення до активної експлуатації та звітування. Кожен крок відображає оперативні рішення, прийняті під час імітованого тесту на проникнення.

Рисунок 2.3 - Візуальне представлення моделі зловмисника

2.2 Планування етапу збору інформації

Етап збору інформації, який також називають розвідкою, є одним з найбільш важливих компонентів будь-якого тесту на проникнення. Він закладає основу для всіх подальших дій, визначаючи і відображаючи компоненти цільової системи, видимі для зовнішнього зловмисника. У тестуванні на проникнення до веб-сторінок точність, глибина та ефективність розвідки безпосередньо впливають на якість аналізу та використання вразливостей.

Для змодельованого зловмисника в цьому дослідженні етап збору інформації ретельно розроблений, щоб відобразити реальні практики і відповідати визнаним стандартам, таким як OWASP WSTG, PTES і NIST SP 800-115. Метою цього етапу є створення розгорнутої та точної карти поверхні атаки на ціль - авторизований домен www.qrngua.website.

Для цього використовується збалансоване поєднання пасивних та активних методів розвідки з використанням спеціального інструментарію, зокрема Subfinder, HTTPX, Naabu, Nuclei, Dirsearch, Waymore та Acunetix. Ці інструменти дозволяють ідентифікувати субдомени, живі сервіси, компоненти додатків, кінцеві точки та потенційні слабкі місця як на рівні інфраструктури, так і на рівні веб-додатків.

Планування процесу розвідки включає в себе:

- Вибір інструментів, які максимізують покриття та мінімізують шум
- Стратегічне поєднання пасивних та активних методів
- Підготовка даних для автоматизованого сканування та ручного аналізу

Кожен інструмент і метод обирається не ізольовано, а як частина цілісного робочого процесу, що відображає поведінку кваліфікованих суб'єктів загроз і етичних хакерів. Крім того, етапи попередньої та подальшої обробки гарантують, що в конвеєр оцінки вразливостей потрапляють лише чисті, релевантні та придатні для вжиття заходів дані.

2.2.1 Загальний огляд розвідки в контексті пентестування веб-додатків

Розвідка - це початкова фаза тестування на проникнення веб-додатків, під час якої зловмисник збирає якомога більше інформації про цільову систему, не намагаючись безпосередньо її використати. Ця фаза має вирішальне значення, оскільки від глибини і точності зібраної на ній інформації залежить ефективність всіх наступних фаз - сканування, аналізу вразливостей і експлуатації.

У контексті цього дослідження розвідка використовується для побудови детального профілю поверхні атаки авторизованої цілі, www.qrngua.website. Зловмисник прагне розкрити:

- Усі доступні субдомени, IP-адреси та службові порти
- Кінцеві точки, параметри та шляхи в Інтернеті
- Основні технології, фреймворки та деталі CMS
- Потенційні точки входу, такі як форми входу, панелі завантаження, API
- Неправильні конфігурації та розголошення інформації з HTTP-заголовків, SSL/TLS тощо.

Ця інформація збирається як пасивними (ненав'язливими), так і активними (інтерактивними) методами. Пасивна розвідка може включати використання DNS-пошуку, журналів прозорості сертифікатів і сторонніх джерел, таких як SecurityTrails. Активні методи передбачають надсилання запитів до цілі (наприклад, через HTTPX або Feroxbuster) для виявлення живих систем, сканування портів і перерахування каталогів.

Розвідка - це не одноразове завдання, а ітеративний процес. При виявленні нових субдоменів або каталогів вони рекурсивно перевіряються. Цей етап вимагає балансу між охопленням і непомітністю: занадто агресивний підхід може викликати контроль безпеки, тоді як занадто пасивний може пропустити цінну інформацію.

Відповідно до таких фреймоворків, як OWASP WSTG, MITRE ATT&CK (T1595 - активне сканування) і PTES, розвідка слугує кільком стратегічним цілям:

- Зменшити сліпі зони в зоні тестування
- Керування автоматизованими інструментами, такими як Nuclei та Acunetix, шляхом попереднього заповнення правильних мішеней
- Підтримка ручної експлуатації (наприклад, виявлення прихованих сторінок входу або неправильно налаштованих каталогів для тестування Burp Suite)

У цьому проекті розвідка повністю інтегрована в методологію зловмисника і виконується за допомогою автоматизованої системи, яка забезпечує узгодженість, відтворюваність і повне відстеження сесій для звітності та аналізу.

2.2.2 Вибір інструменту для автоматичної розвідки

У сучасному тестуванні на проникнення до веб-додатків вибір інструментів розвідки має бути стратегічно обґрунтованим і відповідати як цілям змодельованого зловмисника, так і особливостям архітектури цільового середовища. Інструменти повинні не просто виконувати окремі задачі, а логічно доповнювати один одного, формуючи цілісну розвідку — від початкового виявлення DNS-записів і субдоменів до аналізу відкритих портів, структури каталогів, технологічного стеку та вразливих ендпоінтів.

Для реалізації даного підходу у цьому дослідженні були відібрані надійні, перевірені часом інструменти наведені у таблиці Г.2 у Додатку Г, які демонструють високу продуктивність, підтримуються активною спільнотою, постійно оновлюються та легко інтегруються в автоматизовані сценарії сканування. Їх комбінація дозволяє досягти високого покриття поверхні атаки, зменшити надмірність даних та забезпечити глибину розвідки. Усі інструменти разом

утворюють модульний стек розвідки, який відповідає підходам, що застосовуються сучасними кіберзлочинцями і командами наступальної безпеки.

Кожен інструмент був відібраний не лише за індивідуальними характеристиками, але й за здатністю безперешкодно інтегруватися в ланцюговий робочий процес, де вихідні дані одного інструменту безпосередньо впливають на наступний. Цей модульний і скриптовий конвеєр дозволяє проводити швидку, повторювану і масштабовану розвідку широкого спектру цільових поверхонь.

Цей стратегічний вибір інструментів забезпечує високе покриття, мінімальну надмірність та ефективне використання ресурсів, що відповідає моделям поведінки зловмисників, описаним в MITRE ATT&CK, OSSTMM та NIST SP 800-115.

2.2.3 Пасивні та активні методи розвідки

Розвідку в тестуванні на проникнення можна розділити на два взаємодоповнюючих підходи: пасивний і активний. Кожна методика має свої переваги та ризики, а при спільному використанні вони максимізують охоплення виявлення, балансуючи між непомітністю та точністю.

Пасивна розвідка - це процес збору інформації про цільову систему без прямого контакту з нею. Цей метод використовується для уникнення виявлення системами запобігання вторгненням (IPS/WAF) і не генерує логів на об'єкті.

Основні методи пасивної розвідки включають перерахування DNS на основі сторонніх баз даних (наприклад, за допомогою Subfinder у поєднанні з SecurityTrails), отримання інформації з SSL/TLS-сертифікатів через журнали прозорості, вилучення історичних URL-адрес та метаданих з таких джерел, як Wayback Machine або Common Crawl, аналіз JavaScript-файлів на предмет відкритих ендпоінтів, а також виявлення витоків коду або конфігурацій у відкритих репозиторіях (наприклад, GitHub). Додатково можуть застосовуватись пошукові запити до WHOIS та ASN для збору технічної інформації про цільову інфраструктуру.

Активна розвідка передбачає пряму взаємодію з цільовою системою — надсилання HTTP-запитів, зондування сервісів і сканування контенту. Вона є більш інформативною, але водночас має більший ризик бути виявленою захисними

системами. До типових активних методів належать HTTP-зондування (наприклад, через HTTPX) для визначення статусів відповідей та технологій, сканування TCP-портів за допомогою Naabu, перебір каталогів і файлів інструментом Dirsearch, а також глибокий скриптовий аналіз JavaScript-файлів з метою виявлення ендпоінтів, ключів доступу або конфігурацій.

Наступна діаграма (рис. 2.4) ілюструє, як пасивні та активні методи розвідки логічно розділені, але обидва походять із спільного етапу розвідки. Кожна гілка включає інструменти та типи даних, що використовуються в цій структурі:

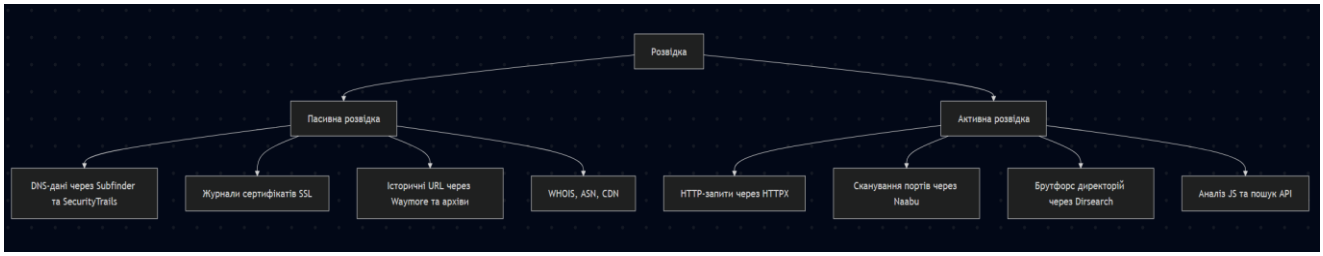


Рисунок 2.4 – Різновидність розвідки

У цьому дослідженні обидва типи розвідки об'єднані та послідовні:

- Пасивні методи використовуються спочатку для мінімізації шуму та створення базової інвентаризації активів.
- Активні методи перевіряють і розширюють результати, готуючи чисті дані для інструментів сканування вразливостей, таких як Nuclei та Acunetix.

Цей гібридний підхід відображає найкращі практики методологій випробувань на проникнення, таких як PTES, OWASP WSTG та MITRE ATT&CK (T1595), гарантуючи, що жоден важливий елемент поверхні цілі не буде пропущений, зберігаючи при цьому операційний реалізм.

2.2.4 Попередня обробка та постобробка даних розвідки

Інформація, зібрана під час розвідки, може бути обширною, суперечливою і частково надлишковою. Щоб забезпечити її надійність і придатність для використання на наступних етапах випробування на проникнення, вона повинна пройти структуровані етапи попередньої і подальшої обробки. Ці етапи є життєво важливими для створення чистого, дедуплікованого і збагаченого набору даних, придатного для сканування, використання і звітування.

Попередня обробка відбувається одразу після збору початкових даних за допомогою таких інструментів, як Subfinder, HTTPX, Naabu та Waymore. Вона включає наступні дії:

1) Нормалізація форматів. Різні інструменти виводять результати у різних форматах (наприклад, JSON, звичайний текст, CSV). Усі записи стандартизовано до єдиного формату, як правило, це списки з переходом на новий рядок або словники JSON, щоб забезпечити сумісність з наступними інструментами.
2) Дедуплікація та фільтрація. Видаляються дублікати субдоменів, IP-адрес або URL-адрес. Домени, про які відомо, що вони належать не пов'язаним CDN, припаркованим службам або зовнішнім інтеграціям, відфільтровуються для зменшення шуму.
3) Валідація живих хостів. Пасивні знахідки перевіряються через HTTPX, щоб переконатися, що кінцеві точки доступні та реагують. Для подальшого сканування зберігаються тільки живі цілі.
4) Сортування та маркування. Хости тегуються на основі таких ознак, як наявність форм входу, шаблонів API, кодів статусу або підписів технологій (наприклад, WordPress, PHP, Nginx).
5) Маркування сеансів. Кожен набір даних позначається міткою часу та ідентифікатором сеансу для відстеження та відтворення під час ланцюжка інструментів та звітності.

Постобробка. Після активного сканування (наприклад, за допомогою Nuclei або Acunetix), постобробка фокусується на перетворенні результатів у структурований і придатний для використання звіт:

1) Кореляція результатів. Результати, отримані різними інструментами, об'єднуються. Наприклад, якщо і Nuclei, і Acunetix виявляють неправильно сконфігурований заголовок на одній кінцевій точці, вони записуються як один об'єднаний результат.
2) Категоризація за ступенем тяжкості. Кожному результату присвоюється рівень пріоритету (наприклад, критичний, високий, середній, низький) на

основі заздалегідь визначених показників CVSS або оцінок для конкретного інструменту.

3) Усунення помилкових спрацьовувань. Ручна перевірка за допомогою Burp Suite та логічного огляду допомагає підтвердити легітимність вразливостей перед включенням до фінального звіту.
4) Підготовка до експорту. Остаточні структуровані результати (хост, проблема, ризик, опис, корисне навантаження, етапи відтворення) експортуються у форматах, сумісних зі звітністю (наприклад, Markdown, HTML або PDF).
5) Архівування сесій. Журнали, результати роботи інструментів, скріншоти і скрипти зберігаються в папках для кожного сеансу, формуючи повний криміналістичний слід процесу тестування.

Така систематична обробка даних розвідки гарантує, що тест на проникнення є технічно точним і відтворюваним, забезпечуючи надійну основу для виявлення вразливостей та аналізу ризиків. Він також відображає найкращі практики, викладені в таких стандартах, як NIST SP 800-115 та OWASP Testing Guide, які підкреслюють необхідність структурованої документації та цілісності даних протягом усього життєвого циклу тестування.

2.3 Огляд архітектури фреймворку

Складність і масштаб сучасних веб-середовищ вимагають, щоб розвідка та аналіз вразливостей були не лише ретельними, але й автоматизованими, повторюваними та узгодженими. Щоб досягти цього, під час дослідження було розроблено спеціальний фреймворк для координації роботи декількох інструментів, управління даними сеансів і створення структурованих результатів на різних етапах тесту на проникнення.

Цей фреймворк відображає основну методологію симульованого зловмисника, представлену раніше: поєднання легких інструментів з відкритим вихідним кодом для швидкості та широти охоплення з цільовим комерційним скануванням для глибини та точності. Кожен інструмент відіграє певну роль у конвеєрі, а їх виконання контролюється централізованою системою на основі

Python з вбудованими можливостями управління конфігурацією, ведення журналів і відстеження сеансів.

Характеристики архітектури фреймворку:

- Модульний: Кожен інструмент може виконуватися незалежно або як частина заздалегідь визначеного ланцюжка
- Масштабований: Легко розширюється на нові домени, інструменти або вихідні формати
- Конфігурованим: Використовує .ini файли для ключів API, профілів сканування та параметрів
- Простежується: Зберігає метадані сеансу та результати для кожного циклу тестування

Цей підрозділ містить детальний огляд внутрішньої логіки та процесу виконання фреймворку, який лежить в основі всіх етапів практичного тестування, описаних далі в цій роботі.

2.3.1 Огляд логіки конвеєра

Конвеєр, що лежить в основі розробленого фреймворку, призначений для відтворення робочого процесу реального зловмисника, який починається з виявлення і закінчується перевіркою та структурованим звітуванням. Цей конвеєр об'єднує як пасивні, так і активні інструменти розвідки, сканери вразливостей та компоненти звітності в поетапний потік виконання, кожна фаза якого ґрунтується на попередній.

Загальний потік автоматизованого конвеєра виглядає наступним чином:

$$Subfinder \rightarrow HTTPX \rightarrow Naabu \rightarrow Waymore \rightarrow Dirsearch \rightarrow Nuclei \rightarrow Acunetix$$

Кожен інструмент виконується послідовно з ретельно структурованою логікою, яка забезпечує чисті вхідні дані, обмежені операції та узгоджені результати. Конвеєр налаштований на роботу в повністю автоматизованому режимі, але також підтримує ручне перевизначення та автономне виконання будь-якого компонента з метою тестування або налагодження.

Конвеєр роботи фреймворку починається з виконання інструмента Subfinder, який здійснює перерахування субдоменів. Він звертається до кількох пасивних джерел, зокрема через API SecurityTrails, щоб виявити як відомі, так і менш очевидні субдомени цільового ресурсу.

Наступним кроком є HTTPX, який отримує на вхід результати Subfinder і перевіряє активність кожного хоста в реальному часі. При цьому фіксуються коди відповідей, HTTP-заголовки та визначаються технології, що використовуються на сервері.

Після цього Naabu сканує живі хости для виявлення відкритих TCP-портів і доступних сервісів. Це дозволяє розширити уявлення про інфраструктуру та ідентифікувати нестандартні або небезпечні веб-інтерфейси.

Далі виконується Waymore, який без активного трафіку витягує кінцеві точки, URL-адреси та інші потенційно корисні елементи з публічних джерел і JavaScript-коду. Це забезпечує глибшу розвідку структури веб-додатку.

Отримані результати передаються до Dirsearch, який проводить активний перебір каталогів і файлів. Таким чином виявляються приховані або незахищені ресурси, резервні копії та допоміжні панелі адміністрування.

Після цього на основі отриманих URL-адрес запускається Nuclei. Він здійснює сканування на вразливості, використовуючи шаблони CVE, неправильної конфігурації або відсутніх безпекових заголовків.

Завершальним етапом є Acunetix, який застосовується для поглибленого аналізу вибраних цілей. Він дозволяє виявити вразливості, пов’язані з логікою бізнес-процесів, обробкою сесій, автентифікацією та іншими складними аспектами, які потребують авторизації або інтерактивної перевірки.

Цей конвеєр виконується в автоматизованому скрипті на основі Python, який обробляє:

- Виконання та послідовність інструментів
- Синтаксичний аналіз і перетворення вхідних/вихідних даних
- Відстеження сесій та очищення тимчасових файлів

- Умовна логіка (наприклад, якщо немає живих хазяїв → пропустити наступні етапи)

Автоматизована система забезпечує відтворюваність, швидкість і точність, зберігаючи при цьому гнучкість для тестувальника, щоб підключити ручні етапи (наприклад, валідацію Burp Suite), де це необхідно.

2.3.2 Обробка даних та типи виводу

Ефективне тестування на проникнення вимагає не тільки виконання інструментів, але й структурованого управління їхніми результатами. У розробленій системі дані з кожної фази пайплайну систематично зберігаються у папці для кожного сеансу, названій на честь основного цільового домену (наприклад, qrngua.website). Це забезпечує чітку, узгоджену структуру для всіх тестових сесій і полегшує доступ, перегляд і звітування.

Кожен вихідний файл наведений у таблиці Г.3 у Додатку Г називається відповідно до інструменту-джерела та його функції в конвеєрі. Ці файли або використовуються як вхідні дані для наступних етапів, або слугують частиною остаточних доказів для звітності.

Основні моменти фреймворка

- Вихідні дані кожного інструменту передаються як вхідні дані для наступного, що гарантує відсутність необхідності ручного втручання під час стандартної роботи.
- Всі дії інструменту реєструються, а часові мітки файлів зберігаються для підтримки перевірки результатів і кореляції в часі.
- Фреймворк очищає тимчасові файли після кожного запуску, зберігаючи всі відповідні результати в каталозі домену.

Ця структурована модель обробки даних відображає рекомендації таких стандартів, як NIST SP 800-115 і OWASP WSTG, які наголошують на чіткому веденні записів, відстежуваності та модульному тестуванні. Вона також сприяє плавному переходу до етапів ручної експлуатації (наприклад, через Burp Suite) або звітування, забезпечуючи при цьому відтворюваність у майбутніх циклах випробувань або аудитів.

2.3.3 Стратегія ланцюжка інструментів

Ефективність створеної на замовлення системи полягає в її здатності координувати низку відкритих і комерційних інструментів безпеки в логічно пов'язаний автоматизований ланцюжок. Такий підхід відображає поведінку реальних зловмисників і гарантує, що кожен етап оцінки ґрунтується на перевірених і відфільтрованих даних попереднього етапу. Це дозволяє проводити автоматизовані масові оцінки, зберігаючи при цьому деталізацію і відстежуваність, необхідну для глибокої ручної перевірки.

Принципи проєктування фреймворку базуються на модульному та файловому підході, що забезпечує гнучкість, прозорість і розширюваність. Кожен інструмент у ланцюжку працює незалежно, зберігаючи свої результати у визначений файл (наприклад, subs.txt, alive.txt тощо). Наступний етап зчитує ці дані як вхідні, що дозволяє організувати послідовну логіку обробки без необхідності дублювання операцій. Якщо будь-який проміжний файл відсутній або порожній, скрипт автоматично припиняє виконання залежного етапу, що дозволяє уникнути зайвих обчислень та потенційних помилок у наступних діях.

Така архітектура має низку переваг: по-перше, зберігається прозорість, оскільки усі проміжні результати фіксуються; по-друге, забезпечується можливість відлагодження, адже кожен крок можна відтворити та перевірити незалежно; по-третє, фреймворк легко масштабується та модифікується, адже окремі компоненти можна замінювати або додавати нові без перегляду всієї логіки.

Послідовність етапів обробки виглядає так: спочатку Subfinder виявляє піддомени й виводить їх у subs.txt; потім HTTPX перевіряє їх активність і формує файл alive.txt. Далі Naabu сканує відкриті порти та зберігає дані у naabu_ports.txt. Інструмент Waymore виконує пасивне збирання кінцевих точок і записує результати в waymore_url.txt. Потім Dirsearch проводить активний перебір каталогів і формує окремі звіти для кожного домену. Nuclei запускає шаблони вразливостей на живих хостах, а результат зберігається у nuclei_out.txt. Нарешті, Acunetix виконує поглиблене сканування й створює фінальний звіт у форматі PDF або HTML (acunetix_report.pdf). Важливо, що кожен крок виконується лише за

умови успішного виконання попереднього, що забезпечує стабільність і контроль виконання.

З точки зору реалізації, вся логіка фреймворку написана на Python з використанням модуля subprocess. Налаштування інструментів та параметри зберігаються у конфігураційному файлі config.ini. Система автоматично створює окремі каталоги для кожного сеансу сканування, генерує логи, мітки часу та дозволяє запуск як повного ланцюга, так і окремих етапів. Крім того, передбачено підтримку аргументів для вибіркового виконання, обробки конкретних діапазонів рядків у списках доменів і вказівки шляхів до вхідних/вихідних файлів.

На практиці одна команда терміналу (наприклад, python run.py -i subfinder--file domains.txt --scan full) може запустити весь процес від розвідки до звітування.

Частковий приклад того, як керується ланцюжок і відстежуються імена вихідних файлів, представлено в Додатку А, де функція run_htsub() координує Subfinder і HTTPX, тоді як інші блоки обробляють сканування Naabu, Nuclei і Acunetix на основі прямого введення.

Фреймворк також керує взаємодією API Acunetix, включаючи:

- Створення та налаштування цілей
- Ініціювання та моніторинг сканування
- Формування та завантаження звіту

Ці частини реалізації наведені у Додатку Б, який містить анотований код Python, що відповідає за безпечний зв'язок з Acunetix через його REST API, включаючи автентифікацію на основі токенів та обробку профілів сканування.

Стратегічна та методологічна актуальність розробленого ланцюгового підходу полягає в тому, що він послідовно реалізує багаторівневу розвідку, яка логічно переходить у фазу експлуатації вразливостей. Така побудова відповідає принципам, закладеним у провідних методологіях тестування безпеки. Зокрема, структура фреймворку узгоджується з підходом OWASP WSTG, де кожен тест базується на результатах попереднього, підтримуючи ідею наскрізної перевірки та валідації. Вона також резонує з методологіями PTES і OSSTMM, які наголошують

на важливості поступового виявлення загроз і побудови довіри на основі фактичних метрик.

Крім того, використання модульної побудови фреймворку прямо відображає концепції MITRE ATT&CK, зокрема при переході від тактики T1595 (Active Scanning) до T1580 (Web Session Cookie Discovery), демонструючи логічну ескалацію атак. Запропонований підхід також враховує сучасні практики DevSecOps — інтеграцію безпекових інструментів у CI/CD-конвеєри, а також автоматизацію розвідки, що використовується в червоних командах, баг-баунті-програмах і регулярному моніторингу вразливостей у реальному часі.

2.4 Висновки за розділом

У цьому розділі було розроблено чітку концептуальну та технічну основу для практичної частини дослідження. Було сформовано модель зловмисника, яка відображає реальні дії атакуючого у сценарії «чорного ящика», та визначено цільовий об’єкт тестування — веб-ресурс QRNG. Застосування методів STRIDE, DREAD та MITRE ATT&CK дозволило здійснити структуроване моделювання загроз з акцентом на ймовірні вектори атак.

Було обґрунтовано вибір інструментів розвідки, таких як Subfinder, HTTPX, Naabu, Waymore, Dirsearch, Nuclei та Acunetix, які формують модульний стек і логічно поєднуються в ланцюговий фреймворк. Застосовано гібридну стратегію, яка поєднує пасивні й активні методи збору даних, з подальшою попередньою обробкою, фільтрацією та постобробкою результатів. Також було спроєктовано власну архітектуру фреймворку на Python, яка автоматизує процес виявлення, класифікації та сканування вразливостей, забезпечуючи повторюваність, масштабованість та інтеграцію з інструментами оцінки.

3 ПРАКТИЧНА РЕАЛІЗАЦІЯ, ТЕСТУВАННЯ ТА АНАЛІЗ РЕЗУЛЬТАТІВ

3.1 Реалізація та розгортання фреймворку

Для успішного розгортання фреймворку для автоматизованого сканування безпеки необхідно ретельно виконати наступні детальні кроки підготовки:

Налаштування API Acunetix: Створіть чітко структурований файл config.ini, що містить URL-адресу кінцевої точки Acunetix API та ключ API. Кожна конфігурація повинна зберігатися під ідентифікованим ім'ям екземпляра для зручності пошуку під час сканування.

Приклад заповненого файлу config.ini наведено на рисунку 3.1.

```
[acunetix_localhost]
url = https://localhost:3443/api/v1/
api=API_KEY_HERE
```

Рисунок 3.1 – Приклад заповненого файлу config.ini

Можна отримати необхідний Acunetix API ключ перейшовши до налаштувань за допомогою веб-інтерфейсу: https://localhost:3443/app/profile.

Для того, щоб уникнути можливих проблем зі з’єднаням потрібно переконатись, що API ключ та URL-адреса є точними и активними.

Встановлення та перевірка інструментів:

Потрібно, щоб був установлений кожен з наступних інструментів:

- Subfinder [19].
- Httpx [20].
- Naabu [28].
- Waymore [22].
- Dirsearch [26].
- Nuclei [21].

Після встановлення потрібно перевірити кожен інструмент окремо, запустивши відповідні команди --help або -h у вашому терміналі, щоб переконатися у правильності інсталяції та конфігурації. Крім того, переконатись, що кожен

інструмент додано до змінної PATH системи для ефективного доступу до нього з командного рядка.

Підготовка необхідних вхідних даних є важливим етапом перед запуском фреймворку. Передусім необхідно чітко визначити цільові домени: це може бути або один домен, переданий безпосередньо через командний рядок, або список доменів, збережений у текстовому файлі, де кожен домен розміщується в окремому рядку.

Також слід завчасно обрати відповідний профіль сканування в Acunetix залежно від цілей безпекової перевірки. Доступні варіанти включають: high — для швидкого пріоритетного тестування, sql — для виявлення SQL-ін’єкцій, та full — для повноцінного глибокого сканування.

Крім того, варто визначити формат вихідного звіту. Acunetix підтримує формати PDF і HTML, тож обирати слід з урахуванням ваших вимог до подальшого аналізу або презентації результатів.

3.1.1 Приклад запуску фреймворку

Нижче (рис. 3.2) наведено детальний приклад виконання всього конвеєра сканування безпеки, включаючи перерахування субдоменів, перевірку активних хостів, комплексне сканування портів, детальне виявлення URL-адрес, автоматичне сканування вразливостей та створення PDF-звітів за допомогою Acunetix:

```
python framework.py -i acunetix_localhost -u qrngua.website --run -s full -r pdf
```

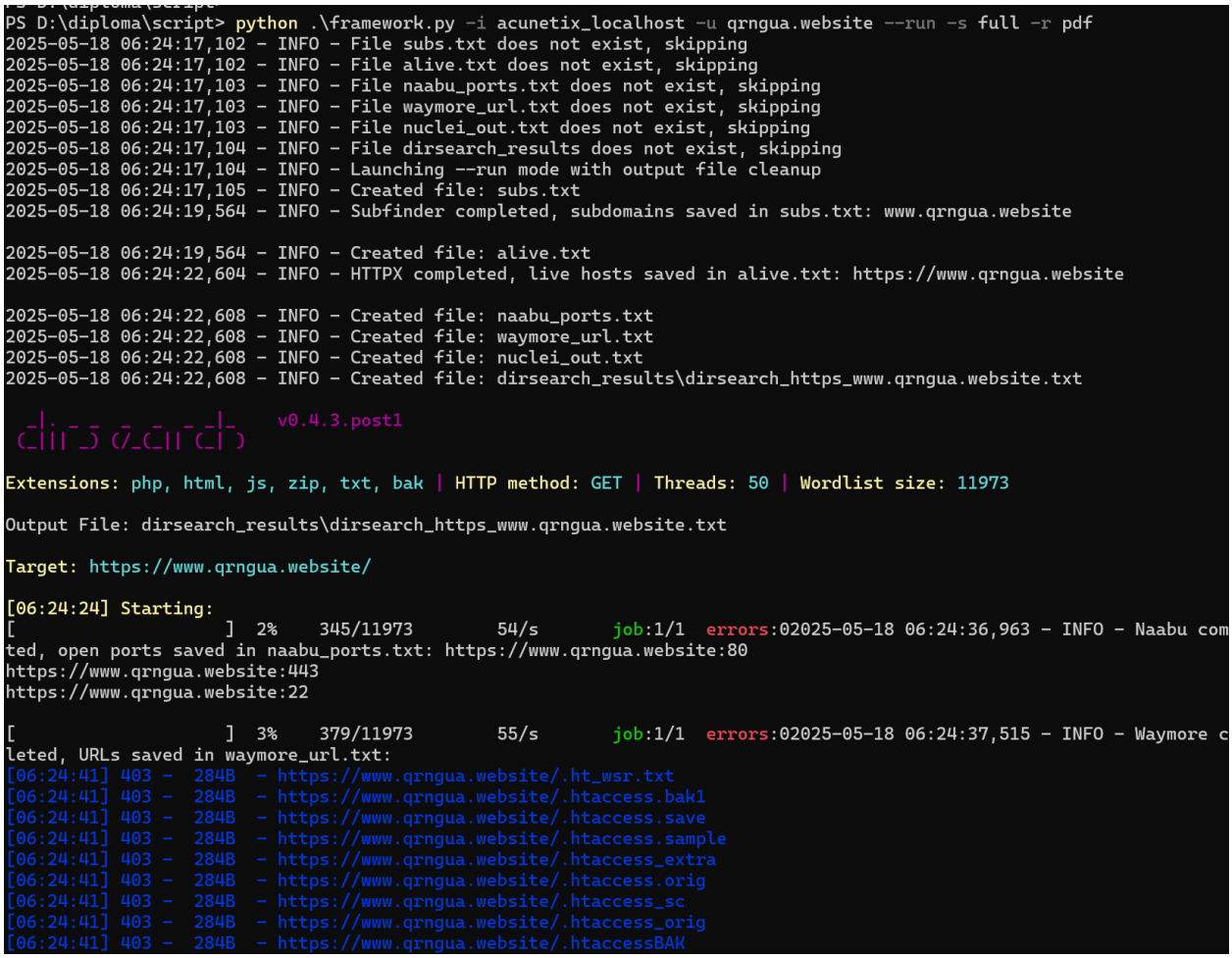


Рисунок 3.2 - Запуск нашого фреймворку (у повному функціоналі)

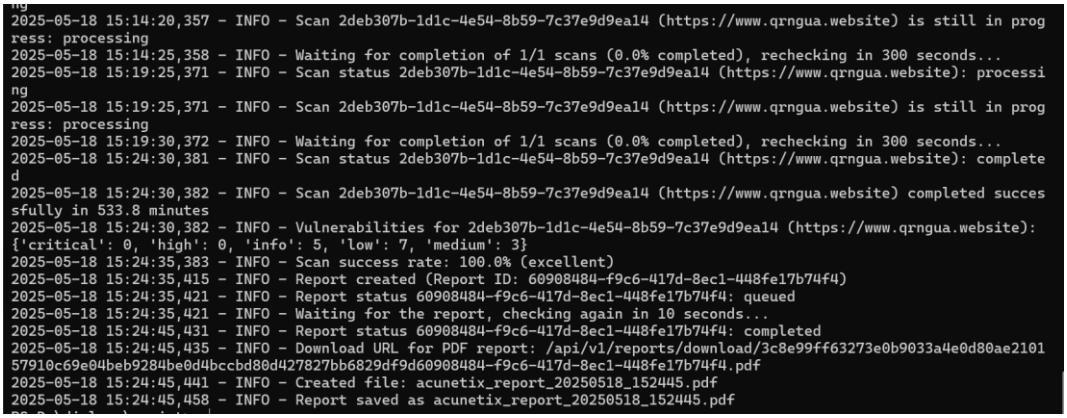


Рисунок 3.3 - Отримання статусу сканування за допомогою Acunetix API та автоматичне створення звіту

Детальне пояснення аргументів команди:

- -i: Вказує ім'я екземпляра Acunetix, визначене у файлі config.ini.
- -u: Вказує цільовий домен для аналізу.

- --run: Ініціює повну послідовність виконання конвеєра із залученням усіх інтегрованих інструментів безпеки.
- -s: Визначає вибраний профіль сканування Acunetix(full,high,sql).
- -r: Визначає бажаний формат виведення звіту (pdf,html).

Самостійно виконувати перерахування піддоменів, використовуючи тільки Subfinder (рис 3.4):

```
python framework.py --subfinder -u qrngua.website --subfinder-output subs.txt
```

```
PS D:\diploma\script> python .\framework.py --subfinder -u qrngua.website --subfinder-output quss.txt
2025-05-18 16:47:30,370 - INFO - Created file: quss.txt
2025-05-18 16:47:32,841 - INFO - Subfinder completed, subdomains saved in quss.txt: www.qrngua.website
```

Рисунок 3.4 – Вибірковий запуск одного інструмента (subfinder) по нашій цілі qrngua.website

Команда заначено на рисунку 3.4 перераховує виключно субдомени для вказаного домену (qrngua.website), зберігаючи всі виявлені субдомени у файлі subs.txt.

1) Основні аргументи:
    - Конфігурація Acunetix : -i, -s, -r
    - Передаванная цілі: -u, -f
    - Запуск повноцінного фреймворку: --run
2) Виборичні запуски певних інструментів:
    - Розвідувальні інструменти та сканери : --subfinder, --httpx, --naabu, --waymore, --dirsearch, --nuclei
3) Додаткові аргументи:
    - Керування конкретними рядками для обробки у великих доменних файлах за допомогою -t (початковий рядок) та -e (кінцевий рядок).
    - Вказуйте власні шляхи до вхідних і вихідних файлів, щоб точно контролювати область застосування та вихідні дані окремих інструментів сканування (--httpx-output alive.txt, --subfinder-output subs.txt).

Дотримання цього детального алгоритму забезпечить надійне розгортання, точне виконання та ефективну роботу вашої комплексної системи оцінки безпеки.

3.2 Аналіз знайдених вразливостей

Під час автоматизованого сканування веб-ресурсу за допомогою інструменту Acunetix (рис. 3.5) було виявлено низку вразливостей, які умовно поділяються на три категорії: середнього, низького рівня серйозності та інформаційного характеру.

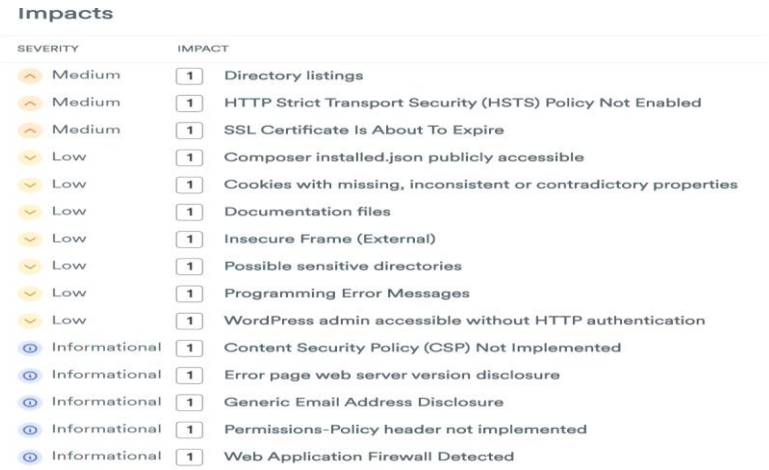
Серед вразливостей середнього рівня найбільш критичною є відкритий доступ до каталогів (Directory Listings Exposure) — такі шляхи, як /wp-content/uploads, /wp-includes/ та /wp-admin/, були доступні без належного контролю, що створює ризик несанкціонованого ознайомлення зі структурою або вмістом сайту. Другою проблемою стала відсутність політики HTTP Strict Transport Security (HSTS), яка підвищує ймовірність атак типу «людина посередині» (MITM), особливо при доступі через незахищене з'єднання. Також було зафіксовано наближення дати завершення дії SSL-сертифіката, що може спричинити помилки безпеки під час встановлення HTTPS-з’єднання.

До вразливостей низького рівня належать кілька конфігураційних і архітектурних недоліків. Наприклад, файл installed.json, що містить інформацію про залежності Composer, був доступний у відкритому доступі. Файли cookie виявилися некоректно налаштованими — відсутній атрибут SameSite, що може дозволити здійснення міжсайтових атак (CSRF). Крім цього, виявлені публічні файли документації (readme.txt, license.txt) можуть допомогти зловмиснику визначити версії компонентів застосунку. Додатково спостерігалися повідомлення про помилки, які можуть розкривати внутрішню інформацію, та відсутній захист до адміністративного інтерфейсу WordPress, що дозволяє доступ до нього без обмежень.

У розділі інформаційних вразливостей зафіксовано відсутність заголовка Content Security Policy (CSP), що підвищує ризик XSS-атак. Також було виявлено розкриття версії веб-сервера у повідомленнях про помилки, що може допомогти зловмиснику у виборі відповідного експлойту. Публічні адреси електронної пошти, знайдені на сторінках, потенційно піддаються фішинговим кампаніям. Відсутність заголовків Permissions-Policy створює додаткову поверхню для зловживання

функціоналом браузера, а наявність брандмауера Wordfence свідчить про активну політику захисту, що, однак, впливає на методи подальшого аналізу.

Усі виявлені недоліки не є критичними самі по собі, проте у сукупності можуть становити суттєвий ризик безпеці веб-ресурсу, особливо за наявності додаткових векторів атаки.

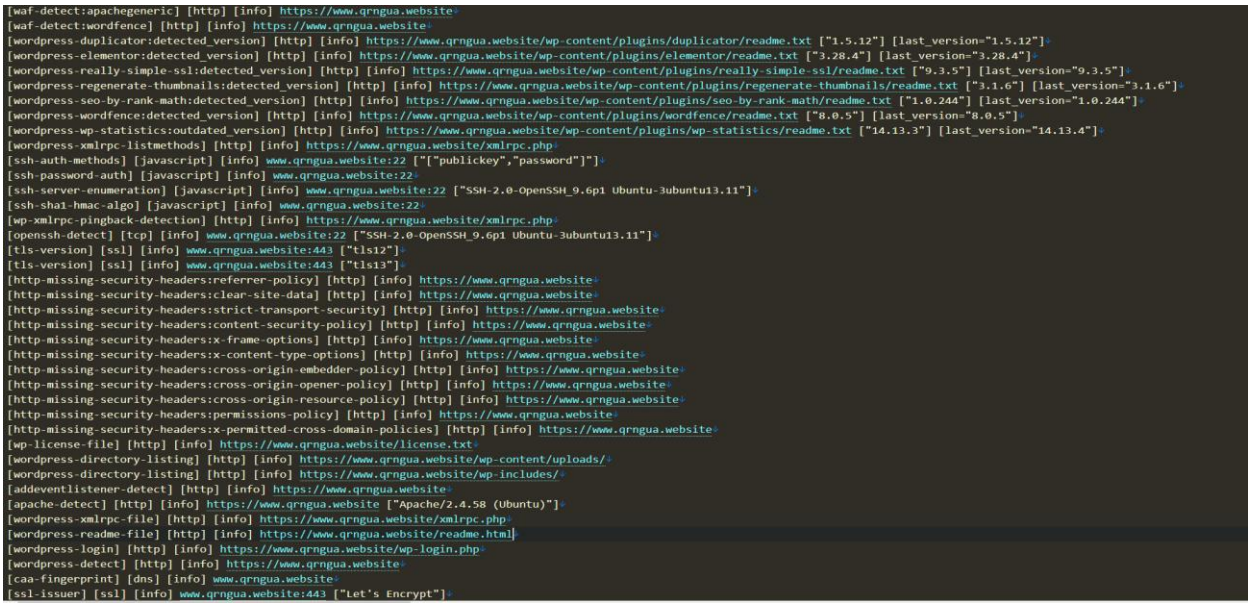


Impacts

SEVERITY	IMPACT	
Medium	1	Directory listings
Medium	1	HTTP Strict Transport Security (HSTS) Policy Not Enabled
Medium	1	SSL Certificate Is About To Expire
Low	1	Composer installed.json publicly accessible
Low	1	Cookies with missing, inconsistent or contradictory properties
Low	1	Documentation files
Low	1	Insecure Frame (External)
Low	1	Possible sensitive directories
Low	1	Programming Error Messages
Low	1	WordPress admin accessible without HTTP authentication
Informational	1	Content Security Policy (CSP) Not Implemented
Informational	1	Error page web server version disclosure
Informational	1	Generic Email Address Disclosure
Informational	1	Permissions-Policy header not implemented
Informational	1	Web Application Firewall Detected

Рисунок 3.5 – Звіт з інструменту Acunetix

Знахідки Nuclei наведено на риснку 3.6. Виявлено застарілі плагіни WordPress, такі як WP Statistics, версія ssh, версія Apache, сторінка входу в WordPress та кінцеві точки, такі як xmlrpc.php. Підтверджено захист брандмауера веб-додатків Wordfence і виявлено кілька відсутніх критично важливих заголовків безпеки, включаючи Content-Security-Policy, X-Frame-Options, Directory Listing і Referrer-Policy.



```
[waf-detect:apachegeneric] [http] [info] https://www.qrngua.website
[waf-detect:wordfence] [http] [info] https://www.qrngua.website
[wordpress-duplicator:detected_version] [http] [info] https://www.qrngua.website/wp-content/plugins/duplicator/readme.txt ["1.5.12"] [last_version="1.5.12"]
[wordpress-elementor:detected_version] [http] [info] https://www.qrngua.website/wp-content/plugins/elementor/readme.txt ["3.28.4"] [last_version="3.28.4"]
[wordpress-really-simple-ssl:detected_version] [http] [info] https://www.qrngua.website/wp-content/plugins/really-simple-ssl/readme.txt ["9.3.5"] [last_version="9.3.5"]
[wordpress-regenerate-thumbnails:detected_version] [http] [info] https://www.qrngua.website/wp-content/plugins/regenerate-thumbnails/readme.txt ["3.1.6"] [last_version="3.1.6"]
[wordpress-seo-by-rank-math:detected_version] [http] [info] https://www.qrngua.website/wp-content/plugins/seo-by-rank-math/readme.txt ["1.0.244"] [last_version="1.0.244"]
[wordpress-wordfence:detected_version] [http] [info] https://www.qrngua.website/wp-content/plugins/wordfence/readme.txt ["8.0.5"] [last_version="8.0.5"]
[wordpress-wp-statistics:outdated_version] [http] [info] https://www.qrngua.website/wp-content/plugins/wp-statistics/readme.txt ["14.13.3"] [last_version="14.13.4"]
[wordpress-xmlrpc-listmethods] [http] [info] https://www.qrngua.website/xmlrpc.php
[ssh-auth-methods] [javascript] [info] www.qrngua.website:22 ["["publickey","password"]"]
[ssh-password-auth] [javascript] [info] www.qrngua.website:22
[ssh-server-enumeration] [javascript] [info] www.qrngua.website:22 ["SSH-2.0-OpenSSH_9.6p1 Ubuntu-3ubuntu13.11"]
[ssh-sha1-hmac-algo] [javascript] [info] www.qrngua.website:22
[wp-xmlrpc-pingback-detection] [http] [info] https://www.qrngua.website/xmlrpc.php
[openssh-detect] [tcp] [info] www.qrngua.website:22 ["SSH-2.0-OpenSSH_9.6p1 Ubuntu-3ubuntu13.11"]
[tls-version] [ssl] [info] www.qrngua.website:443 ["tls12"]
[tls-version] [ssl] [info] www.qrngua.website:443 ["tls13"]
[http-missing-security-headers:referrer-policy] [http] [info] https://www.qrngua.website
[http-missing-security-headers:clear-site-data] [http] [info] https://www.qrngua.website
[http-missing-security-headers:strict-transport-security] [http] [info] https://www.qrngua.website
[http-missing-security-headers:content-security-policy] [http] [info] https://www.qrngua.website
[http-missing-security-headers:x-frame-options] [http] [info] https://www.qrngua.website
[http-missing-security-headers:x-content-type-options] [http] [info] https://www.qrngua.website
[http-missing-security-headers:cross-origin-embedder-policy] [http] [info] https://www.qrngua.website
[http-missing-security-headers:cross-origin-opener-policy] [http] [info] https://www.qrngua.website
[http-missing-security-headers:cross-origin-resource-policy] [http] [info] https://www.qrngua.website
[http-missing-security-headers:permissions-policy] [http] [info] https://www.qrngua.website
[http-missing-security-headers:x-permitted-cross-domain-policies] [http] [info] https://www.qrngua.website
[wp-license-file] [http] [info] https://www.qrngua.website/license.txt
[wordpress-directory-listing] [http] [info] https://www.qrngua.website/wp-content/uploads/
[wordpress-directory-listing] [http] [info] https://www.qrngua.website/wp-includes/
[addeventlistener-detect] [http] [info] https://www.qrngua.website
[apache-detect] [http] [info] https://www.qrngua.website ["Apache/2.4.58 (Ubuntu)"]
[wordpress-xmlrpc-file] [http] [info] https://www.qrngua.website/xmlrpc.php
[wordpress-readme-file] [http] [info] https://www.qrngua.website/readme.html
[wordpress-login] [http] [info] https://www.qrngua.website/wp-login.php
[wordpress-detect] [http] [info] https://www.qrngua.website
[caa-fingerprint] [dns] [info] www.qrngua.website
[ssl-issuer] [ssl] [info] www.qrngua.website:443 ["Let's Encrypt"]
```

Рисунок 3.6 – Вивід інструменту Nuclei

Знахідки Naabu. Знайдено відкриті порти:

- Port 22 (SSH) виявлено, що збільшує ризик несанкціонованого доступу та атак грубої сили(brute force).
- Port 80 (HTTP) виявлено, що дозволяє потенційно незахищену комунікацію.
- Port 443 (HTTPS) перевірено наявність безпечного з'єднання, але виявлено проблеми з сертифікатом SSL з Acunetix.

Знахідки Dirsearch наведено на рисунку 3.7. Виявлено загальнодоступні конфіденційні файли та каталоги, включаючи резервні копії конфігурації (wp-config.php), інформацію про ліцензію (license.txt) та скрипти, пов'язані з адмініструванням (wp-cron.php, xmlrpc.php).

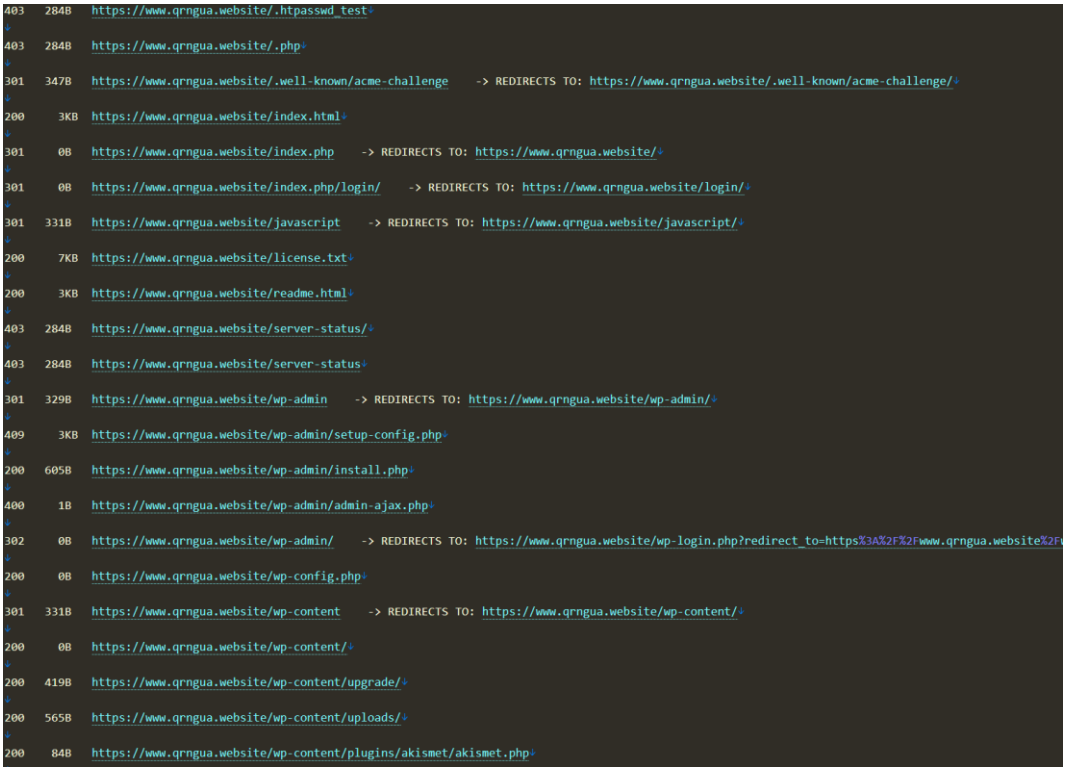


Рисунок 3.7 - Частковий вивід інструменту Dirsearch

На рисунку 3.7 зображено: код відповіді від сервера (200, 403, 301), довжину відповіді, посилання на файл чи папку.

3.2.1 Валідація, перевірка, експлуатація вручну

Acunetix отримав 0 критичних, 0 високих, 3 середніх, 7 низьких та 5 інформативних вразливостей.

1) Лістинг файлів у директоріях:

https://www.qrngua.website/wp-content/plugins/robo-gallery/includes/extensions/block/dist/

https://www.qrngua.website/wp-content/plugins/duplicator/

https://www.qrngua.website/wp-includes/

https://www.qrngua.website/wp-includes/css/

https://www.qrngua.website/wp-includes/css/dist/

https://www.qrngua.website/wp-includes/css/dist/block-editor/

https://www.qrngua.website/wp-content/plugins/wp-statistics/assets/js/

https://www.qrngua.website/wp-admin/images/

https://www.qrngua.website/wp-admin/includes/

https://www.qrngua.website/wp-content/uploads/

https://www.qrngua.website/wp-admin/js/

Кожна папка, у наведеному вище лістингу, була виявлена за допомогою перевірки слова Index of {folder}.

Переглянувши кожну папку на рисунку 3.8, та кожен файл, не було знайдено ніяких доступних паролів чи іншої корисної інформації для експлуатації. Не проводилось аналізу php коду так як він не доступний для перегляду, було знайдено pdf файли у директорії [/wp-content/uploads/2025/01] (рис. 3.9) та [/wp-content/uploads/2024/12/] (рис. 3.10).

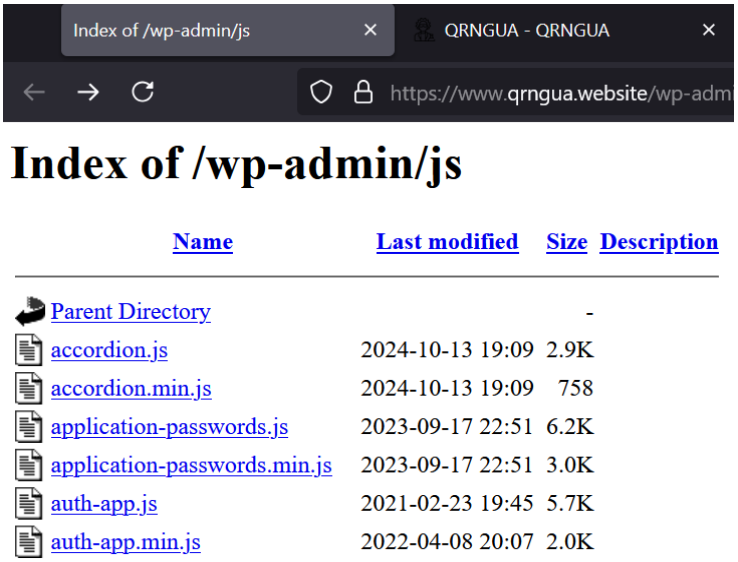


Рисунок 3.8 - Лістинг директорії на прикладі [/wp-admin/js]

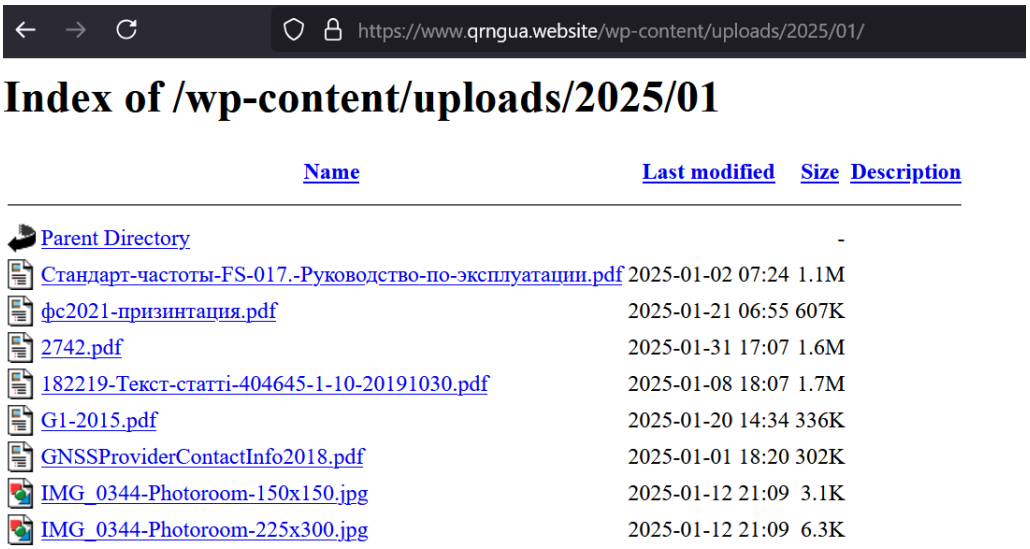


Рисунок 3.9 – Знайдені PDF – файли [/wp-content/uploads/2025/01]

Рисунок 3.10 – Знайдені PDF – файли [/wp-content/uploads/2024/12/]

Такого типу вразливість може видавати різноманітні документи, які не доступні для перегляду неавторизованим або непривілейованим користувачам, тому слід подбати про те, щоб зловмисник не зміг отримати доступ до різноманітних чутливих файлів.

2) HTTP Strict Transport Security (HSTS) Політика не включена:

HTTP Strict Transport Security (HSTS) повідомляє браузеру, що веб-сайт доступний лише за допомогою HTTPS. Було виявлено, що ваш веб-додаток не використовує (HSTS), оскільки у відповіді відсутній заголовок Strict Transport Security.

HSTS можна використовувати для запобігання та/або пом'якшення наслідків деяких типів атак типу «людина посередині» (MitM). На повних ресурсах (рис. 3.11) політика не ввімкнена.

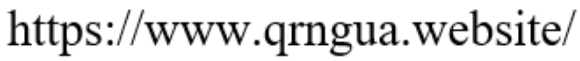


Рисунок 3.11 – Посилання на ресурс

3) SSL Certicate Is About To Expire:

Термін дії одного з сертифікатів TLS/SSL, що використовується вашим сервером, закінчується. Після закінчення терміну дії сертифіката більшість веб-браузерів видають кінцевим користувачам попередження про безпеку з проханням вручну підтвердити автентичність ланцюжка сертифікатів. Програмне забезпечення або автоматизовані системи можуть мовчки відмовити в підключенні до сервера. Це попередження не обов'язково спричинене сертифікатом сервера (листовим), але може бути викликане проміжним сертифікатом.

Якщо сервер додатків виявляє прострочений сертифікат у системі, з якою він взаємодіє, сервер додатків може продовжувати обробляти дані, ніби нічого не сталося, або з'єднання може бути раптово розірвано (рис 3.12).

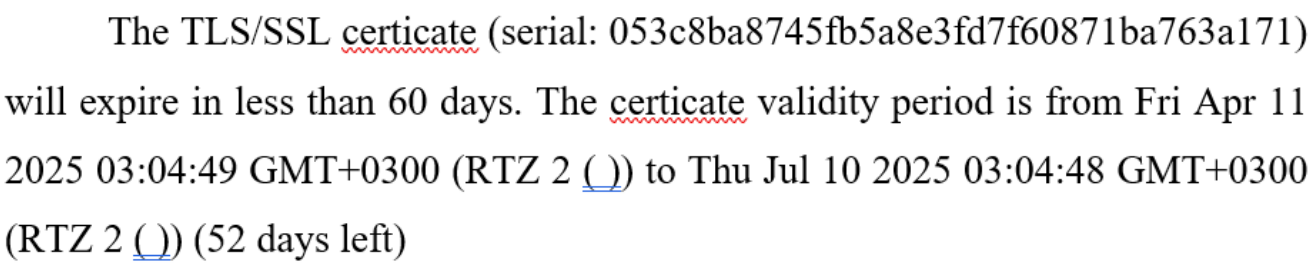


Рисунок 3.12 – Повідомлення попередження

4) Composer installed.json загальнодоступний:

Виявлено installed.json. (рис. 3.13) Composer - це інструмент для управління залежностями в PHP. Він дозволяє вам оголосити бібліотеки, від яких залежить ваш проект, і він буде керувати ними (встановлювати/оновлювати) за вас. Після встановлення залежностей Composer зберігає їх список у спеціальному файлі для внутрішніх цілей.

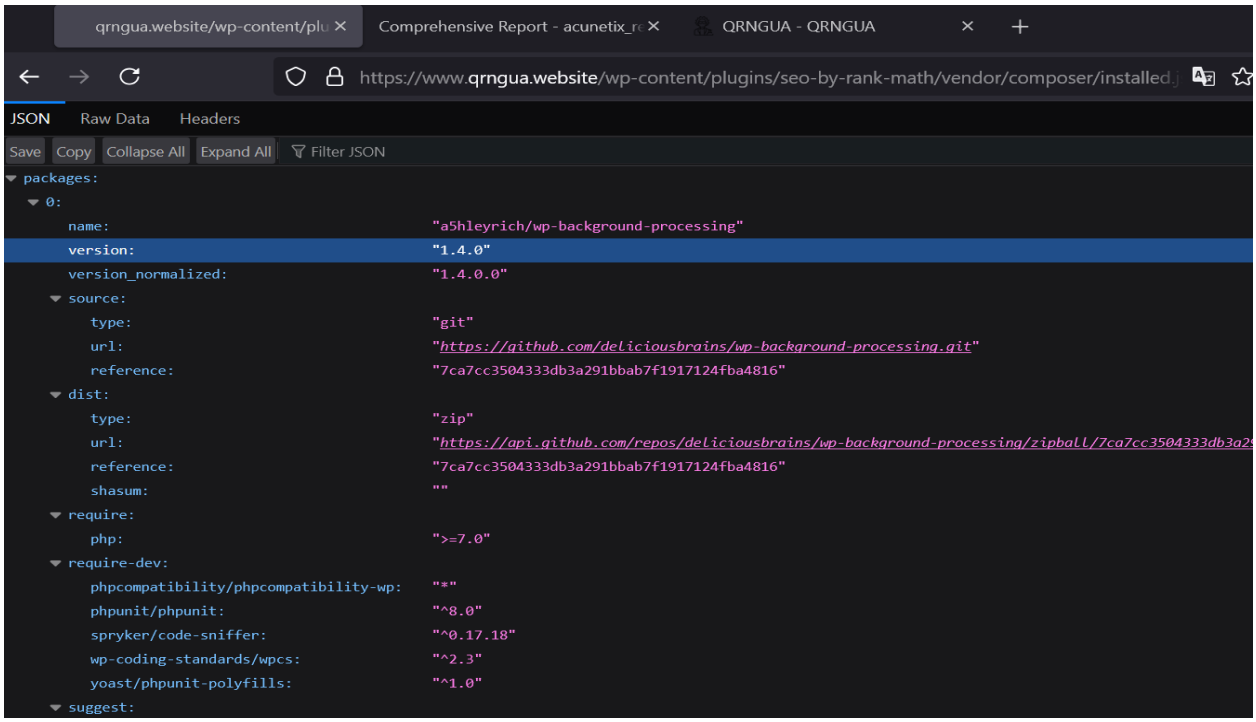


Рисунок 3.13 – installed.json з вимогами до них

5) Файли cookie з відсутніми, непослідовними або суперечливими властивостями:

Принаймні одна з наведених нижче на рисунку 3.14 властивостей файлу cookie робить його недійсним або несумісним з іншими властивостями того ж файлу cookie або з середовищем, в якому він використовується. Хоча це не є вразливістю саме по собі, це може призвести до неочікуваної поведінки програми, що, в свою чергу, може спричинити вторинні проблеми з безпекою.

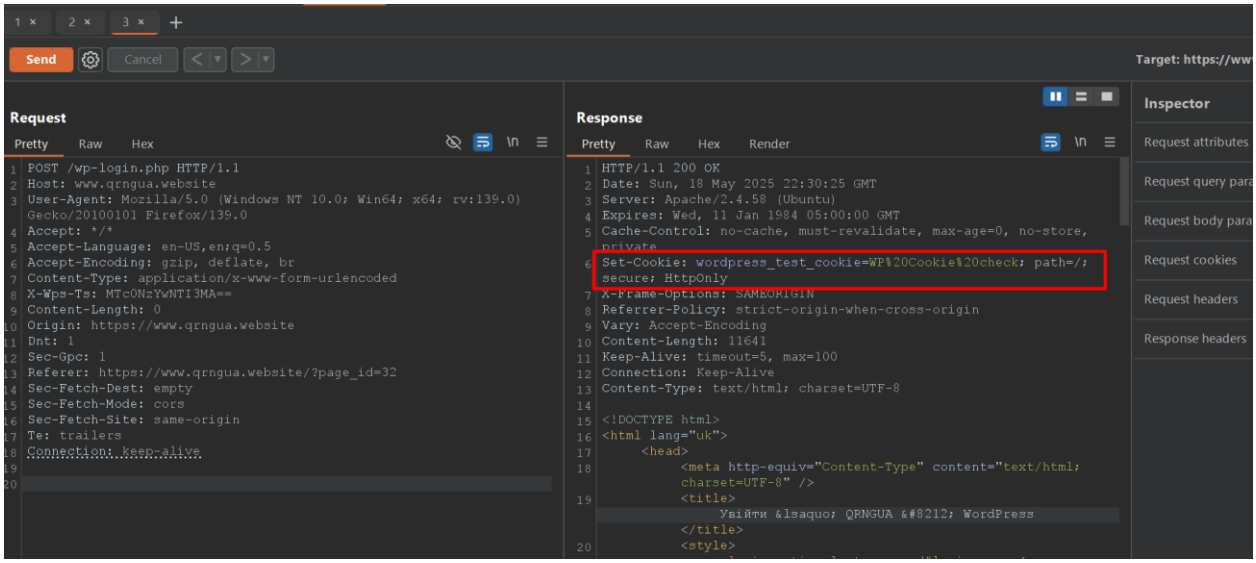


Рисунок 3.14 – PoC Cookie без атрибуту SameSite

Якщо у файлах cookie відсутній атрибут SameSite, веб-браузери можуть застосовувати різні, іноді неочікувані значення за замовчуванням. Тому рекомендується додати атрибут SameSite з відповідним значенням «Strict», “Lax” або «None».

6) Файли документації:

Знайдено файли документації (наприклад, readme.txt, changelog.txt). Інформація, що міститься в цих файлах, може допомогти зловмиснику ідентифікувати веб-додаток, який ви використовуєте, а іноді і його версію. Рекомендується видалити ці файли з робочих систем.

https://www.qrngua.website/wp-content/plugins/robo-gallery/readme.txt

https://www.qrngua.website/wp-content/plugins/duplicator/readme.txt

https://www.qrngua.website/wp-content/plugins/wp-statistics/readme.txt

https://www.qrngua.website/wp-content/plugins/wp-statistics/license.txt

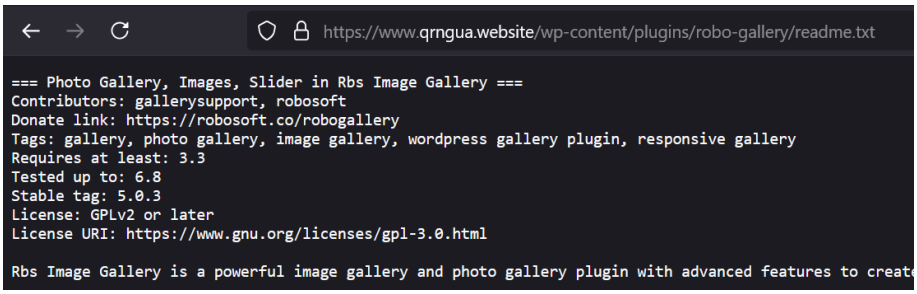


Рисунок 3.15 – PoC Розкриття версії

Зловмисник може використовувати її для пошуку вразливостей у версіях програмного забезпечення. Ця інформація може бути використана для запуску подальших атак.

7) Розкриття помилок у роботі додатку:

Вразливість наведена на рисунку 3.16 є False Positive. Acunetix використовує ключове слово Fatal Error для її виявлення, але там є лише журнал змін плагіна. Отже, вразливість не підтверджена.

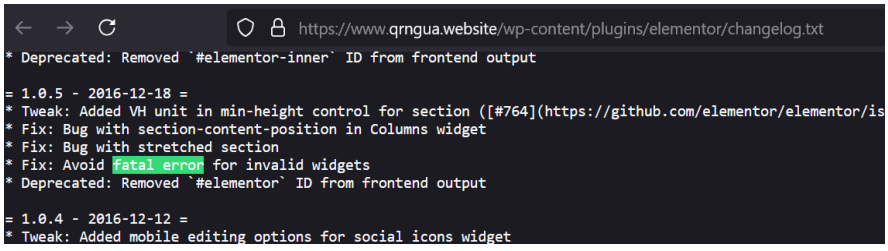


Рисунок 3.16 – Помилкове спрацьовування вразливості.

8) Доступ до адмін панелі WordPress без HTTP-аутентифікації:

Рекомендується обмежити доступ до адміністративної панелі WordPress (рис 3.17 - 3.18) за допомогою HTTP-аутентифікації. Захист паролем адміністративної панелі WordPress за допомогою HTTP-автентифікації є ефективним заходом для запобігання спробам зловмисників вгадати паролі користувачів.

Крім того, якщо зловмисникам вдасться викрасти пароль користувача, їм потрібно буде пройти HTTP-автентифікацію, щоб отримати доступ до форми входу в WordPress.

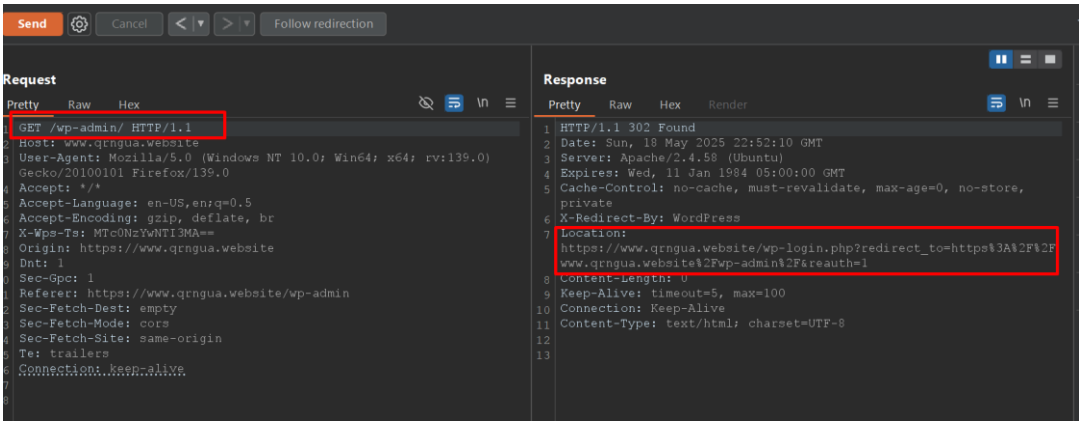
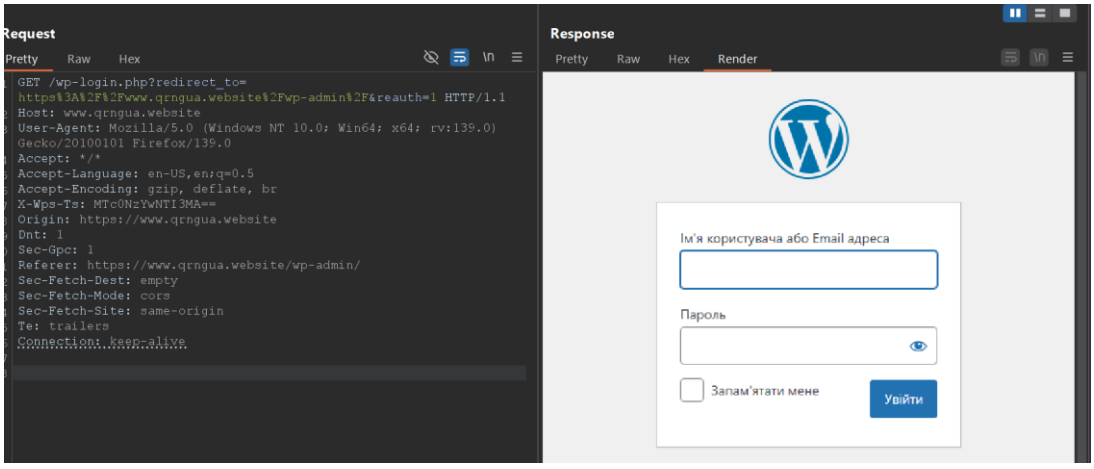


Рисунок 3.17 – Перенаправлення для входу на сторінку адміністратора WordPress



Рисуеок 3.18 – Після переходу з'явилась сторінка входу для адміністратора

9) Інформація про версію веб-сервера на сторінці помилки:

Помилки програми або попереджувальні повідомлення можуть розкрити зловмиснику конфіденційну інформацію про внутрішню роботу програми. Acunetix знайшов номер версії веб-сервера, що ннаведено на рисунку 3.19, та

список модулів, увімкнених на цільовому сервері. Зверніться до розділу «Деталі атаки» для отримання додаткової інформації про уражену сторінку.

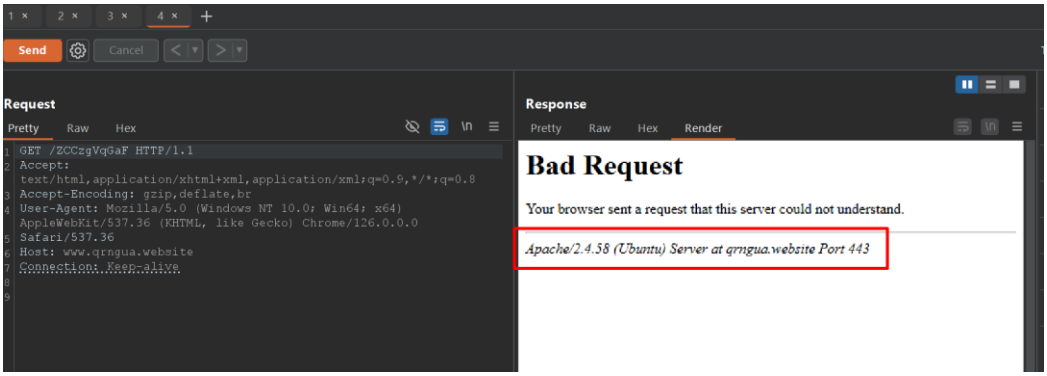


Рисунок 3.19 – Інформація про версію сервера Apache

3.3 Класифікація та оцінка вразливостей

3.3.1 Категоризація на основі CWE

Вразливості, виявлені в результаті автоматизованої та ручної оцінки, були класифіковані відповідно до стандартів Common Weakness Enumeration (CWE), які надають детальну категоризацію та пояснення кожної вразливості:

1) Directory Listing Exposure CWE-548: витік інформації через список каталогів відбувається, коли веб-сервер розкриває вміст каталогів без відповідних обмежень доступу.
2) Missing HTTP Strict Transport Security (HSTS) CWE-319: передача конфіденційної інформації відкритим текстом є наслідком нездатності забезпечити безпечне з'єднання (HTTPS), що підвищує вразливість до перехоплення та MITM-атак.
3) SSL Certificate Expiration CWE-298: неправильна перевірка терміну дії сертифіката означає, що ви не вчасно оновлюєте або перевіряєте SSL-сертифікати, що може призвести до порушення безпечного зв'язку.
4) Public Composer Configuration File Exposure CWE-200: витік інформації відбувається, коли конфіденційні файли, призначені для внутрішнього використання, стають загальнодоступними, розкриваючи конфігурації та залежності додатків.
5) Cookie Misconfiguration (Missing SameSite Attribute) CWE-1275: неправильне обмеження призначення маркерів безпеки вказує на недостатньо захищені

файли cookie, що може сприяти міжсайтовим атакам на підробку запитів (CSRF).

6) Exposed Documentation Files CWE-200: витік інформації відбувається, коли конфіденційна документація, така як readme або файли журналу змін, стає загальнодоступною, розкриваючи критичні деталі про версії та структуру програм.
7) Insecure Iframe Configuration CWE-1021: неправильне обмеження шарів або фреймів візуалізованого інтерфейсу вказує на нездатність безпечно налаштувати елементи iframe, що уможливлює потенційні маніпуляції з боку зовнішніх суб'єктів.
8) Unprotected WordPress Administrative Interfaces CWE-306: відсутність автентифікації для критичної функції відображає відсутність належних заходів автентифікації для доступу до адміністративних компонентів, що призводить до ризику несанкціонованого доступу та підвищення привілеїв.
9) Missing Content Security Policy (CSP) CWE-693: відмова механізму захисту - це відсутність або недостатня реалізація CSP, що уможливлює потенційні міжсайтовий скриптинг (XSS) та подібні атаки.
10) Server Version Disclosure via Error Messages CWE-200: вразливість інформації стосується веб-серверів, які розкривають детальну інформацію про версію через сторінки помилок, що полегшує проведення цілеспрямованих атак.

3.3.2 Оцінка серйозності на основі CVSS

Для визначення серйозності кожної з виявлених вразливостей було використано «Загальну систему оцінки вразливостей» (CVSS v3.1). Детальні оцінки CVSS для кожної вразливості наведені нижче:

1) Directory Listing Exposure CVSS Base Score: 5.3 (Середній)
2) Missing HTTP Strict Transport Security (HSTS) CVSS Base Score: 6.5 (Середній)
3) SSL Certificate Expiration CVSS Base Score: 4.7 (Середній)
4) Public Composer Configuration File Exposure CVSS Base Score: 5.3 (Середній)

5) Cookie Misconfiguration (Missing SameSite Attribute) CVSS Base Score: 3.7 (Низький)
6) Exposed Documentation Files CVSS Base Score: 5.3 (Середній)
7) Insecure Iframe Configuration CVSS Base Score: 4.3 (Середній)
8) Unprotected WordPress Administrative Interfaces CVSS Base Score: 6.8 (Середній)
9) Missing Content Security Policy (CSP) CVSS Base Score: 5.8 (Середній)
10) Server Version Disclosure via Error Messages CVSS Base Score: 5.3 (Середній)

Ці показники CVSS дають чітке і практичне уявлення про серйозність і потенційний вплив виявлених вразливостей, що сприяє прийняттю обґрунтованих рішень та ефективному визначенню пріоритетів для заходів з усунення недоліків.

3.4 Висновки за розділом

У розділі була реалізована й протестована побудована раніше модель атаки та автоматизований фреймворк, що дозволило перейти від теоретичної підготовки до практичних дій. Було показано, як інструменти розвідки, сканування та аналізу можуть працювати у злагодженому ланцюгу — від збору субдоменів до побудови детального звіту про вразливості. Створений пайплайн підтвердив свою ефективність: всі основні етапи виконувалися автономно, з чіткою структурою даних і прозорим логуванням.

Під час сканування ресурсу були виявлені як конфігураційні недоліки, так і типові вразливості середнього та низького рівня, зокрема відкриті каталоги, проблеми з cookie, відсутність захисних заголовків та слабкі місця в SSL-конфігурації. Ці результати були ретельно проаналізовані, підтверджені вручну і класифіковані згідно з міжнародними стандартами CWE і CVSS.

Фреймворк дозволив не лише автоматизувати процеси, які зазвичай виконуються вручну, але й створити просту у використанні та масштабовану основу для подальших перевірок інших веб-додатків.

ВИСНОВКИ

У ході виконання роботи було реалізовано повноцінний підхід до оцінки безпеки веб-додатку шляхом моделювання реальних атак. Проведений теоретичний аналіз охоплював сучасні методи тестування, типи пентесту, моделі загроз, а також міжнародні стандарти, зокрема OWASP WSTG, NIST SP 800-115 та OSSTMM. Це дозволило створити міцну методологічну основу для практичної частини.

Було розроблено власний фреймворк на Python, який автоматизує процес збору розвідданих, сканування та генерації звітів. У фреймворк інтегровано кілька сучасних інструментів (Subfinder, HTTPX, Naabu, Dirsearch, Nuclei, Acunetix), що працюють у логічному ланцюжку та забезпечують покриття основних етапів тестування. Така архітектура дозволила провести ефективне та відтворюване сканування ресурсу QRNG.

У результаті тестування були виявлені вразливості середнього та низького рівня, зокрема відкриті каталоги, відсутність HSTS, проблеми з cookie та публічний доступ до конфігураційних файлів. Частину знахідок було підтверджено вручну за допомогою Burp Suite. Вразливості були класифіковані за CWE та оцінені за шкалою CVSS, що дозволило обґрунтовано розставити пріоритети для усунення.

Практична цінність роботи полягає у створенні універсального інструментарію, який можна застосовувати для подальших перевірок безпеки інших веб-додатків. Результати підтверджують доцільність і ефективність модульного підходу до пентесту, а також актуальність регулярного аналізу безпеки з урахуванням реальних векторів атак.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1) Тест на проникнення [Електронний ресурс]. – Режим доступу: https://uk.wikipedia.org/wiki/Тест_на_проникнення.

2) 2021 Data Breach Investigations Report / Verizon [Електронний ресурс]. – Режим доступу: https://www.verizon.com/business/resources/reports/2021-data-breach-investigations-report.pdf.

3) Baloch R. Web Hacking Arsenal: A Practical Guide to Modern Web Pentesting.

4) Hoffman A. Web Application Security: Exploitation and Countermeasures for Modern Web Applications. – 2024.

5) Stuttard D., Pinto M. The Web Application Hacker's Handbook: Finding and Exploiting Security Flaws.

6) OSSTMM 3.0 – Open Source Security Testing Methodology Manual [Електронний ресурс]. – Режим доступу: https://www.isecom.org/OSSTMM.3.pdf.

7) Javier F. Bug Bounty from Scratch: A comprehensive guide to discovering vulnerabilities and succeeding in cybersecurity. – 2024.

8) Onofri S., Onofri D. Attacking and Exploiting Modern Web Applications.

9) OWASP Web Security Testing Guide v4.2 [Електронний ресурс]. – Режим доступу: https://owasp.org/www-project-web-security-testing-guide/v42/.

10) NIST SP 800-115. Technical Guide to Information Security Testing and Assessment [Електронний ресурс]. – Режим доступу: https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-115.pdf.

11) HackTricks. Pentesting Methodology [Електронний ресурс]. – Режим доступу: https://book.hacktricks.wiki/en/generic-methodologies-and-resources/pentesting-methodology.html.

12) RFC 2350 [Електронний ресурс]. – Режим доступу: https://datatracker.ietf.org/doc/html/rfc2350.

13) RFC 9116 [Електронний ресурс]. – Режим доступу: https://datatracker.ietf.org/doc/html/rfc9116.

14) Common Weakness Enumeration (CWE) [Електронний ресурс]. – Режим доступу: https://cwe.mitre.org/.

15) HackerOne Disclosure Guidelines [Електронний ресурс]. – Режим доступу: https://www.hackerone.com/disclosure-guidelines.

16) Vulnerability Disclosure Policy Template [Електронний ресурс]. – Режим доступу: https://www.hackerone.com/resources/disclosure-guidelines-template.

17) ISSAF – Information Systems Security Assessment Framework [Електронний ресурс]. – Режим доступу: https://untrustednetwork.net/files/issaf0.2.1.pdf.

18) Electronics 2023, 12(5), 1229 [Електронний ресурс]. – Режим доступу: https://www.mdpi.com/2079-9292/12/5/1229.

19) Subfinder [Електронний ресурс]. – Режим доступу: https://github.com/projectdiscovery/subfinder.

20) HTTPX [Електронний ресурс]. – Режим доступу: https://github.com/projectdiscovery/httpx.

21) Nuclei [Електронний ресурс]. – Режим доступу: https://github.com/projectdiscovery/nuclei.

22) Waymore [Електронний ресурс]. – Режим доступу: https://github.com/xnl-h4ck3r/waymore.

23) SecurityTrails [Електронний ресурс]. – Режим доступу: https://securitytrails.com/.

24) URLFinder [Електронний ресурс]. – Режим доступу: https://github.com/projectdiscovery/urlfinder.

25) Katana [Електронний ресурс]. – Режим доступу: https://github.com/projectdiscovery/katana.

26) Dirsearch [Електронний ресурс]. – Режим доступу: https://github.com/maurosoria/dirsearch.

27) Acunetix Web Scanner [Електронний ресурс]. – Режим доступу: https://www.acunetix.com/plp/acunetix-scanner/.

28) Naabu [Електронний ресурс]. – Режим доступу: https://github.com/projectdiscovery/naabu.

29) Закон України «Про інформацію» [Електронний ресурс]. – Режим доступу: https://zakon.rada.gov.ua/laws/show/2341-14#Text.

30) Code of Ethics / EC-Council [Електронний ресурс]. – Режим доступу: https://www.eccouncil.org/code-of-ethics/.

31) ISC2 Code of Ethics [Електронний ресурс]. – Режим доступу: https://www.isc2.org/Ethics.

32) Heartland Payment Systems breach [Електронний ресурс]. – Режим доступу: https://www.twingate.com/blog/tips/Heartland%20Payment%20Systems-data-breach.

33) Twitter XSS Worm [Електронний ресурс]. – Режим доступу: https://www.theguardian.com/technology/blog/2010/sep/21/twitter-hack-explained-xss-javascript.

34) Shellshock vulnerability [Електронний ресурс]. – Режим доступу: https://blog.qualys.com/vulnerabilities-threat-research/2014/09/24/bash-remote-code-execution-vulnerability-cve-2014-6271.

35) Equifax Breach [Електронний ресурс]. – Режим доступу: https://investor.equifax.com/news-events/press-releases/detail/237/equifax-releases-details-on-cybersecurity-incident.

36) Avery.com ransomware attack [Електронний ресурс]. – Режим доступу: https://therecord.media/avery-products-ransomware-data-breach-notification.

37) STRIDE Model [Електронний ресурс]. – Режим доступу: https://en.wikipedia.org/wiki/STRIDE_model.

38) DREAD Model [Електронний ресурс]. – Режим доступу: https://en.wikipedia.org/wiki/DREAD_(risk_assessment_model).

39) CIA Triad [Електронний ресурс]. – Режим доступу: https://en.wikipedia.org/wiki/Information_security#CIA_triad.

40) CVSS – Common Vulnerability Scoring System [Електронний ресурс]. – Режим доступу: https://www.first.org/cvss/.

41) MITRE ATT&CK Framework [Електронний ресурс]. – Режим доступу: https://www.ibm.com/think/topics/mitre-attack.

ДОДАТОК А

КОД КЛАСУ SUBDOMAIN_PROBE

```
def run_htsub(domain=None, domain_list=None,
subfinder_output="subs.txt", httpx_output="alive.txt"):
    """Runs Subfinder and HTTPX in a coordinated manner."""
    # Run Subfinder
    if domain:
        run_subfinder(domain=domain, output_file=subfinder_output)
    elif domain_list:
        run_subfinder(domain_list=domain_list,
output_file=subfinder_output)
    else:
        logger.error("Either domain or domain_list must be provided
for run_htsub()")
        return
    # Check for Subfinder output file
    if not ensure_file_exists(subfinder_output):
        logger.error(f"Subfinder did not create expected file
{subfinder_output}")
        return


    # Run HTTPX using Subfinder output
    run_httpx(input_file=subfinder_output, output_file=httpx_output)
```

ДОДАТОК Б

КОД КЛАСУ ACUNETIX_SCAN

```
# Load API configuration from config.ini file
def get_instance_config(instance_name):
    config = load_config()
    if not config or instance_name not in config:
        logger.error(f"Instance '{instance_name}' not found.")
        return None
    return {
        "url": config[instance_name].get("url", ""),
        "api": config[instance_name].get("api", "")
    }
# Setup session with retry logic for robust HTTP requests
def setup_session():
    session = requests.Session()
    retries = Retry(total=3, backoff_factor=1,
status_forcelist=[429, 500, 502, 503, 504])
    session.mount("https://", HTTPAdapter(max_retries=retries))
    return session
# Create a new target in Acunetix
def create_target(api_url, headers, domain, session):
    target_data = {
        "address": domain,
        "description": generate_session_name(),
        "type": "default"
    }
    response = session.post(f"{api_url}targets", headers=headers,
data=json.dumps(target_data), verify=False)
    target_id = response.json().get("target_id")
    return target_id


# Configure the created target with predefined scanning options
```

```
def configure_target(api_url, headers, target_id, session):
    config_data = {
        "scan_speed": "sequential",
        "user_agent": "Mozilla/5.0...",
        "login": {"kind": "none"},
        "ssh_credentials": {"kind": "none"},
        "default_scanning_profile_id": "11111111-1111-1111-1111-
111111111111",
        "sensor": False,
        "limit_crawler_scope": True,
        "authentication": {"enabled": False},
        "proxy": {"enabled": False},
        "ad_blocker": True
    }
    session.patch(f"{api_url}targets/{target_id}/configuration",
headers=headers, data=json.dumps(config_data), verify=False)
# Start a vulnerability scan for the configured target
def start_scan(api_url, headers, target_id, profile_id, session):
    scan_data = {
        "target_id": target_id,
        "profile_id": profile_id,
        "schedule": {"disable": True}
    }
    response = session.post(f"{api_url}scans", headers=headers,
data=json.dumps(scan_data), verify=False)
    scan_id = response.json().get("scan_id")
    return scan_id
# Continuously check scan status until completion
def wait_for_scans(api_url, headers, scan_info, session=None):
    pending_scans = {scan_id: domain for scan_id, domain in
scan_info}
    completed_statuses = ["completed", "failed", "aborting"]
    while pending_scans:
        for scan_id in list(pending_scans):
```

```
            status, domain, _ = check_scan_status(api_url, headers,
scan_id, session)
            if status in completed_statuses:
                del pending_scans[scan_id]
            time.sleep(5)
        time.sleep(300)
# Request Acunetix to generate a report based on completed scans
def create_report(api_url, headers, scan_ids, session):
    data = {
        "template_id": "11111111-1111-1111-1111-111111111126",
        "source": {"list_type": "scans", "id_list": scan_ids}
    }
    response = session.post(f"{api_url}reports", headers=headers,
json=data, verify=False)
    report_id = response.json().get("report_id")
    return report_id
# Retrieve the download URL for the generated report
def get_report_download_url(api_url, headers, report_id,
format_type="pdf", session=None):
    response = session.get(f"{api_url}reports/{report_id}",
headers=headers, verify=False)
    download_urls = response.json().get("download", [])
    for url in download_urls:
        if url.endswith(f".{format_type}"):
            return url
# Download and save the report file locally
def download_report(api_url, download_url, headers,
format_type="pdf", session=None):
    full_url = f"{api_url.rstrip('/api/v1')}{download_url}"
    response = session.get(full_url, headers=headers, verify=False,
stream=True)
    timestamp = datetime.datetime.now().strftime("%Y%m%d_%H%M%S")
    report_path = f"acunetix_report_{timestamp}.{format_type}"
    with open(report_path, "wb") as f:
```

```
for chunk in response.iter_content(chunk_size=8192):
    f.write(chunk)
```

ДОДАТОК В

ТАБЛИЦІ ПЕРШОГО РОЗДІЛУ

Таблиця В.1 – Значущі атаки, які відображають еволюцію використання вразливостей

Рік, та назва атаки	Суть атаки
2008 – Heartland Payment Systems	Понад 100 мільйонів номерів кредитних карток було викрадено за допомогою SQL-ін'єкції [32].
2010 – XSS-черв'як Twitter	Саморозповсюджуване корисне навантаження на JavaScript скомпрометувало сеанси користувачів через відображену XSS-вразливість [33].
2014 – Shellshock	Зловмисники використовували CGI-скрипти для запуску віддаленого виконання коду через уразливість в Bash [34].
2017 – Equifax Breach	Вразливість в Apache Struts призвела до витоку конфіденційних даних 147 мільйонів користувачів [35].
2024 – Avery.com Breach	У період з липня по грудень 2024 року веб-сайт електронної комерції Avery був скомпрометований шкідливим програмним забезпеченням для скрапінгу кредитних карток, впровадженим за допомогою ін'єкції скриптів. Зловмисники зібрали імена, адреси та дані кредитних карток приблизно 67 000 користувачів до виявлення 9 грудня, що також включало в себе активність «ransomware» [36].

Таблиця В.2 – Порівняльна характеристика формального пентесту та «баг-баунті»

Характеристика	Формальний пентест	Баг-баунті програма
Тестувальники	Внутрішня команда або найняті фахівці	Зовнішня спільнота
Тривалість	Фіксовані часові рамки	Постійний, безперервний
Область застосування	Заздалегідь визначена	Широка або напіввизначена
Вид результату	Офіційний звіт, аналіз ризиків	Індивідуальні висновки за допомогою PoC

Відповідність нормативним вимогам	Висока	Від низького до помірного
Модель витрат	Фіксована плата	Оплата за вразливість (варіативна)
Рівень контролю	Високий	Від середнього до низького

Таблиця В.3 – Порівняльна характеристика різновидів пентесту

Характеристика	Зовнішнє оцінювання	Внутрішня оцінка
Роль зловмисника	Аутсайдер (без доступу)	Інсайдер або зловмисник після порушення
Точка входу	Системи з виходом в Інтернет	Внутрішня мережа або VPN
Першочергові цілі	Веб-додатки, брандмауери, DNS, відкриті порти	AD, внутрішні програми, файлообмінники, сегментація
Фокус	Виявлення, автентифікація,периметр	Зловживання привілеями, латеральний рух, довіра
Моделювання ризиків	Пристосовницький хакер	Внутрішня загроза або просунута постійна загроза

Таблиця В.4 - Порівняння компонентів звітності в тестуванні безпеки

Компонент	Опис	Цінність
CVSS	Стандартизований бал (0–10), який базується на векторі атаки, складності, впливі на CIA тощо.	Дає змогу швидко визначити пріоритет виправлення, порівняти ризики між вразливостями.
Proof of Concept (PoC)	Відтворюваний приклад експлуатації вразливості з URL, HTTP-запитами, скріншотами, скриптами.	Підтверджує реальність загрози, полегшує перевірку й повторне тестування.
NIST SP 800-115	Нарративна форма: мета, методика, результати, CVSS, вплив, рекомендації, додатки.	Орієнтована на технічну й управлінську аудиторію, практична для внутрішніх звітів.
OSSTMM	Кількісна форма: метрики довіри, правила тестування, повторювані операційні показники.	Забезпечує точність, відтворюваність і можливість статистичного порівняння рівнів безпеки.

ДОДАТОК Г

ТАБЛИЦІ ДРУГОГО РОЗДІЛУ

Таблиця Г.1 - Порівняльна таблиця: STRIDE проти DREAD проти MITRE ATT&CK

Характеристика	STRIDE	DREAD	MITRE ATT&CK
Тип	Класифікація загроз	Модель оцінки ризику	Модель реальної поведінки атакуючих
Призначення	Визначення типів загроз	Оцінка та пріоритезація вразливостей	Мапінг тактик і технік зловмисників
Структура	6 типів загроз	5 критеріїв (шкала 1–10)	Тактики → Техніки → Підтехніки
Застосування	Аналіз архітектури	Оцінка знайдених вразливостей	Імітація атак, red teaming
Перевага	Простота та швидкість застосування	Дає числову оцінку ризику	Базується на реальних сценаріях атак

Таблиця Г.2 - Характеристика інструментів, використаних у фреймворку

Інструмент	Призначення	Причина вибору	Інтеграція	Роль у фреймворку
Subfinder	Виявлення піддоменів	Швидкий, надійний, підтримує пасивні/активні джерела	DNS, WHOIS	Формує базовий список для зондування
HTTPX	HTTP-зондування, збір заголовків	Високопродуктивний, виявляє технології	Після Subfinder	Відфільтровує активні хости
Naabu	Сканування TCP-портів	Швидкий аналог Nmap, підтримка автоматизації	Результати → HTTPX	Визначає відкриті порти та сервіси
Waymore	Пасивний збір URL, аналіз JS	Виявляє неочевидні ендпоінти	До перебору каталогів	Знаходить приховані або застарілі API
Dirsearch	Перебір каталогів і файлів	Легкий, ефективний, скриптовий	Живі URL з HTTPX	Виявляє адмін-панелі, резервні копії
Nuclei	Сканування шаблонів CVE	Потужний шаблонний двигун (YAML)	Працює з HTTPX	Швидке виявлення вразливостей
Acunetix	Повне сканування безпеки	Підтримка авторизації, логіки, сесій	Після первинної розвідки	Поглиблений аудит вразливостей

Таблиця Г.3 - Стандартні вихідні файли за інструментами

Назва файлу	Опис
subs.txt	Сирі піддомени, знайдені за допомогою Subfinder (включаючи результати SecurityTrails)
alive.txt	Перевірені живі субдомени, ідентифіковані за допомогою HTTPX (відфільтровані з subs.txt)
naabu_ports.txt	Список відкритих портів для кожного хоста, просканованого Naabu (на основі alive.txt)
waymore_url.txt	Зібрані URL-адреси та кінцеві точки від Waymore (для доменів у файлі alive.txt)
dirsearch_https_domain.com.txt	Перебір каталогів і файлів, знайдених Dirsearch для кожного живого домену
nuclei_out.txt	Результати сканування вразливостей, згенеровані Nuclei для всіх активних кінцевих точок
acunetix_report.pdf	Остаточний звіт про вразливості експортовано з Acunetix у форматі PDF або HTML