Міністерство освіти і науки України
Державний університет «Житомирська політехніка»
Факультет інформаційно-комп'ютерних технологій
Кафедра комп'ютерної інженерії та кібербезпеки

ПОЯСНЮВАЛЬНА ЗАПИСКА

до випускної кваліфікаційної роботи бакалавра

на тему: «Проєкт набору завдань за напрямком Web для проведення навчальних кібербезпекових досліджень та CTF-змагань»

Виконав студент 4-го курсу групи КБ-20-1
спеціальності 125 «Кібербезпека»
_____ Артем ЛАЙЧУК
Керівник: завідувач кафедри КІ та КБ, к.т.н., доцент
_____ Андрій ЄФІМЕНКО
Рецензент: професор кафедри КІ та КБ, д.т.н., доцент
_____ Володимир ВОРОТНІКОВ

Житомир – 2024

Державний університет «Житомирська політехніка»
Факультет інформаційно-комп'ютерних технологій
Кафедра комп'ютерної інженерії та кібербезпеки
Спеціальність 125 «Кібербезпека»
Освітня програма «Кібербезпека»

ЗАТВЕРДЖУЮ
Завідувач
кафедри комп'ютерної
інженерії та кібербезпеки
Андрій ЄФІМЕНКО
5 березня 2024 р.

ЗАВДАННЯ
на випускну кваліфікаційну роботу бакалавра

Студент: Лайчук Артем Андрійович

Тема роботи: «Проєкт набору завдань за напрямком Web для проведення навчальних кібербезпекових досліджень та CTF-змагань»

затверджена Наказом університету від 5 березня 2024 р. № 98

Термін здачі студентом закінченої роботи 10 червня 2024 р.

Вихідні дані роботи (зазначається предмет і об'єкт дослідження):

Об'єктом дослідження є процес створення та експлуатації вразливих веб-сайтів. Предметом дослідження є методи та інструменти для створення навчальних вразливих веб-сайтів та розробка завдань на їх основі для навчальних кібербезпекових досліджень та CTF-змагань.

Обладнання та програмне забезпечення:

1. Oracle VM VirtualBox.
2. Ubuntu 22.04, Kali Linux.
3. MySQL 8.0.36.
4. PHP 8.1.2, Python 3.10.12, Flask 2.0.1, Werkzeug 2.0.2, Apache 2.4.52.
5. VSCodium.
6. Інструменти для експлуатації вразливостей: jwt_tool, sqlmap, hashcat, exploit-CVE-2021-22204, Metasploit.

Консультанти з випускної кваліфікаційної роботи із зазначенням розділів, що їх стосуються

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв
1	Єфіменко А.А.	05.03.2024	05.03.2024
2	Єфіменко А.А.	10.04.2024	10.04.2024
3	Єфіменко А.А.	09.05.2024	09.05.2024

Студент _____ Артем ЛАЙЧУК

Керівник _____ Андрій ЄФІМЕНКО

Календарний план

№ з/п	Назви етапів випускної кваліфікаційної роботи	Термін виконання етапів роботи	Примітка
1	Постановка завдання	05.03.2024	Виконано
2	Розробка плану роботи	07.03.2024	Виконано
3	Аналіз процесу розробки Web-додатків та створення завдань для CTF-змагань	09.03.2024 – 09.04.2024	Виконано
5	Проектування веб-додатків та завдань у форматі CTF	11.04.2024 – 08.05.2024	Виконано
6	Розробка вразливих веб-сайтів	10.05.2024 – 20.05.2024	Виконано
7	Тестування розроблених веб-сайтів	22.05.2024	Виконано
8	Експлуатація вразливостей	24.05.2024	Виконано
9	Написання пояснювальної записки	25.05.2024 – 08.06.2024	Виконано
10	Отримання відгуку та рецензій	14.06.2024	Виконано
11	Захист кваліфікаційної роботи		

Студент _____ Артем ЛАЙЧУК

Керівник _____ Андрій ЄФІМЕНКО

АНОТАЦІЯ

Кваліфікаційна робота складається з пояснювальної записки, що містить опис проектування набору завдань за напрямком Web для навчальних кібербезпекових досліджень та CTF-змагань, а також реалізовані в середовищі моделювання кейси їх практичної реалізації.

Текстова частина викладена на 113 сторінках друкованого тексту. Пояснювальна записка має 17 сторінок додатків. Список використаних джерел містить 27 найменувань та займає 2 сторінки. У роботі наведено 120 рисунків. Загальний обсяг роботи – 138 сторінок.

Метою кваліфікаційної роботи є розробка проєкту набору завдань з категорії Web для навчальних кібербезпекових досліджень та CTF-змагань, орієнтованих на студентів та початківців у сфері Web-захисту, що дозволить їм набути необхідних навичок для виявлення, експлуатації вразливостей та їх запобігання у веб-додатках.

Для досягнення поставленої мети були розв'язані наступні завдання:

– Досліджено теоретичні аспекти створення веб-додатків та їх розгортання на веб-серверах.

– Проаналізовано найпоширеніші вразливості у веб-додатках та методи їх захисту.

– Спроектовано набір із чотирьох завдань з детальним описом цих завдань, вразливостей у веб-додатках та методів їх експлуатації за допомогою обраного інструментарію для успішного захоплення прапорів.

– Розроблено та протестовано чотири вразливих веб-сайта з подальшою експлуатацією реалізованих вразливостей та успішним захопленням прапорів

Ключові слова: CTF, ТЕСТУВАННЯ НА ПРОНИКНЕННЯ, WEB-БЕЗПЕКА, XSS, SQL-ІН'ЄКЦІЯ, ЕКСПЛОЙТ, JWT, CVE.

ANNOTATION

The qualification work consists of an explanatory note describing the design of a set of tasks in the area of Web for cybersecurity research and CTF competitions, as well as cases of their practical implementation implemented in the simulation environment.

The text part is presented on 113 pages of printed text. The explanatory note has 17 pages of appendices. The list of references includes 27 titles and occupies 2 pages. The work contains 120 illustrations. The total volume of the work is 138 pages.

The purpose of the qualification work is to develop a draft set of tasks from the Web category for educational cybersecurity research and CTF competitions aimed at students and beginners in the field of Web security, which will allow them to acquire the necessary skills to identify, exploit and prevent vulnerabilities in web applications.

To achieve this goal, the following tasks were solved:

– The theoretical aspects of creating web applications and their deployment on web servers were investigated.

– The most common vulnerabilities in web applications and methods of their protection were analyzed.

– A set of four tasks is designed with a detailed description of these tasks, vulnerabilities in web applications and methods of their exploitation using the selected tools for successful flags capture.

– Four vulnerable websites have been developed and tested, followed by exploitation of the implemented vulnerabilities and successful flagging.

Keywords: CTF, PENETRATION TESTING, WEB-SECURITY, XSS, SQL INJECTION, EXPLOIT, JWT, CVE.

ЗМІСТ

ВСТУП..... 9
РОЗДІЛ 1. ТЕОРЕТИЧНІ АСПЕКТИ СТВОРЕННЯ WEB-ДОДАТКІВ ТА РОЗРОБКИ ЗАВДАНЬ У ФОРМАТІ CAPTURE THE FLAG 11
1.1. Архітектура та основи створення Web-додатків. 11
1.1.1. Клієнт-серверна модель. 11
1.1.2. Фронтенд та бекенд. 14
1.1.3. Веб-сервери та сервери додатків. 17
1.2. Безпека Web-додатків. 20
1.2.1. Поширені вразливості у веб-додатках. 20
1.2.2. Методики захисту веб-додатків. 24
1.2.3. Інструменти для тестування безпеки веб-застосунків. 28
1.3. Розробка завдань Web-категорії для CTF-змагань. 33
1.3.1. Рівні складності завдань категорії Web у форматі захоплення прапора. 33
1.3.2. Вибір категорії та створення завдань. 36
1.3.3. Тестування та валідація завдань. 37
РОЗДІЛ 2. ПРОЕКТУВАННЯ ВЕБ-ДОДАТКІВ ТА НАБОРУ ЗАВДАНЬ 39
2.1. Аналіз вимог до набору завдань. 39
2.1.1. Цільова аудиторія та контекст застосування. 39
2.1.2. Навчальні цілі. 40
2.2. Розробка специфікацій завдань. 41
2.2.1. Проектування та опис завдань. 41
2.2.2. Визначення рівня складності завдань. 49
2.3. Вибір інструментів та технологій для створення веб-сайтів та реалізації завдань. 51
2.3.1. Середовище виконання. 51
2.3.3. Вибір та проектування бази даних. 57
2.3.4. Інструменти, що необхідні учасникам для експлуатації вразливостей. 59

РОЗДІЛ 3. ПРАКТИЧНА РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ СТВОРЕНИХ ВЕБ-ДОДАТКІВ 62
3.1. Підготовка середовища до розробки веб-додатків 62
3.2. Розробка веб-сайту для дослідження пошуку прихованих шляхів та модифікації HTTP-запитів від клієнта до сервера. 71
3.2.1. Створення веб-сайту. 71
3.2.2. Тестування розробленого функціоналу та його експлуатація для захоплення прапору. 76
3.3. Розробка веб-додатку з вразливостями JSON Web Token та Server-Side Template Injection та їх експуатація. 82
3.3.1. Створення веб-сайту. 82
3.3.2. Тестування розробленого функціоналу та експлуатація реалізованих вразливостей. 85
3.4. Розробка веб-додатку з вразливістю CVE-2021-22204 та її експлуатація. .. 94
3.4.1. Створення веб-сайту. 94
3.4.2. Тестування розробленого функціоналу та експлуатація реалізованої вразливості 97
3.5. Розробка веб-додатку з вразливостями SQLi та Cross-Site Scripting, тестування та експлуатація. 104
3.5.1. Створення веб-сайту. 104
3.5.2. Тестування розробленого функціоналу та експлуатація реалізованих вразливостей. 109
ВИСНОВКИ 119
СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ 120
ДОДАТКИ 122

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

HTML - HyperText Markup Language

CSS - Cascading Style Sheets

HTTP - HyperText Transfer Protocol

HTTPS - HyperText Transfer Protocol Secure

CTF - Capture The Flag

URL - Uniform Resource Locator

CGI - Common Gateway Interface

API - Application Programming Interface

XSS - Cross-Site Scripting

SQL - Structured Query Language

SQLi - SQL-Injection

CVE - Common Vulnerabilities and Exposures

SSTI - Server-Side Template Injection

SSRF - Server-Side Request Forgery

CSRF - Cross-Site Request Forgery

RCE - Remote Code Execution

JSON - JavaScript Object Notation

JWT - JSON Web Token

CLI - Command Line Interface

ВСТУП

Створення безпечних веб-додатків має вирішальне значення через конфіденційні дані, які часто ними обробляються. У сучасну цифрову епоху веб-додатки використовуються для різних цілей, наприклад для онлайн-банкінгу, електронної комерції та комунікації, і всі вони включають конфіденційну інформацію. Якщо веб-додаток не є безпечним, це може призвести до серйозних наслідків, таких як крадіжка особистих даних, фінансові втрати та шкода репутації.

За останні роки, згідно з різними статистичними даними, атаки на веб-додатки зросатють із загрозливою швидкістю, тому навчання та підготовка спеціалістів з кібербезпеки є надзвичайно актуальними. Участь у CTF-змаганнях (Capture The Flag) є ефективним методом підготовки спеціалістів з кібербезпеки, особливо студентів та початківців.

Завдання у такому форматі надають учасникам можливість зануритися в реалістичні сценарії атак та захисту, що дозволяє їм краще зрозуміти механізми та методи забезпечення безпеки веб-додатків, тому створення набору завдань з категорії Web для таких змагань допоможе покращити рівень підготовки учасників.

Метою кваліфікаційної роботи є створення набору завдань за напрямком Web для навчальних кібербезпекових досліджень та CTF-змагань, орієнтованих на студентів та початківців у сфері Web-захисту, що дозволить їм набути необхідних навичок для виявлення, експлуатації вразливостей та їх запобігання у веб-додатках.

Визначена мета обумовлює наступні завдання:

− Дослідження теоретичних аспектів розробки веб-додатків.

− Дослідження сучасних методів атак на веб-додатки та способів захисту від них.

− Розробка комплексу завдань з категорії Web для CTF-змагань.

− Тестування розроблених веб-додатків та експлуатація вразливостей.

Об’єктом дослідження є процес проєктування та створення завдань за напрямком Web для проведення CTF-змагань та навчальних практичних кібербезпекових досліджень.

Предметом дослідження є методи та інструменти для створення навчальних вразливих веб-сайтів та розробка завдань на їх основі для навчальних кібербезпекових досліджень та CTF-змагань.

У процесі виконання кваліфікаційної роботи було розроблено набір із чотирьох завдань з категорії Web для CTF-змагань, що включає різноманітні сценарії атак веб-додатків, спрямовані на підвищення рівня практичної підготовки учасників.

РОЗДІЛ 1. ТЕОРЕТИЧНІ АСПЕКТИ СТВОРЕННЯ WEB-ДОДАТКІВ ТА РОЗРОБКИ ЗАВДАНЬ У ФОРМАТІ CAPTURE THE FLAG

1.1. Архітектура та основи створення Web-додатків.

1.1.1. Клієнт-серверна модель.

Основною архітектурною концепцією для більшості сучасних веб-додатків є клієнт-серверна модель. Модель “Клієнт-сервер” — це розподілена структура програми, яка розподіляє завдання або робочі навантаження між постачальниками ресурсів або послуг, які називаються серверами, і запитувачами послуг, які називаються клієнтами [4]. В архітектурі клієнт-сервер, коли клієнтський комп’ютер надсилає запит на дані серверу через Інтернет, сервер приймає запитуваний процес і доставляє запитувані пакети даних назад клієнту. Ця модель зазвичай використовується в різних додатках, включаючи веб-додатки, системи електронної пошти, файлові сервери тощо.

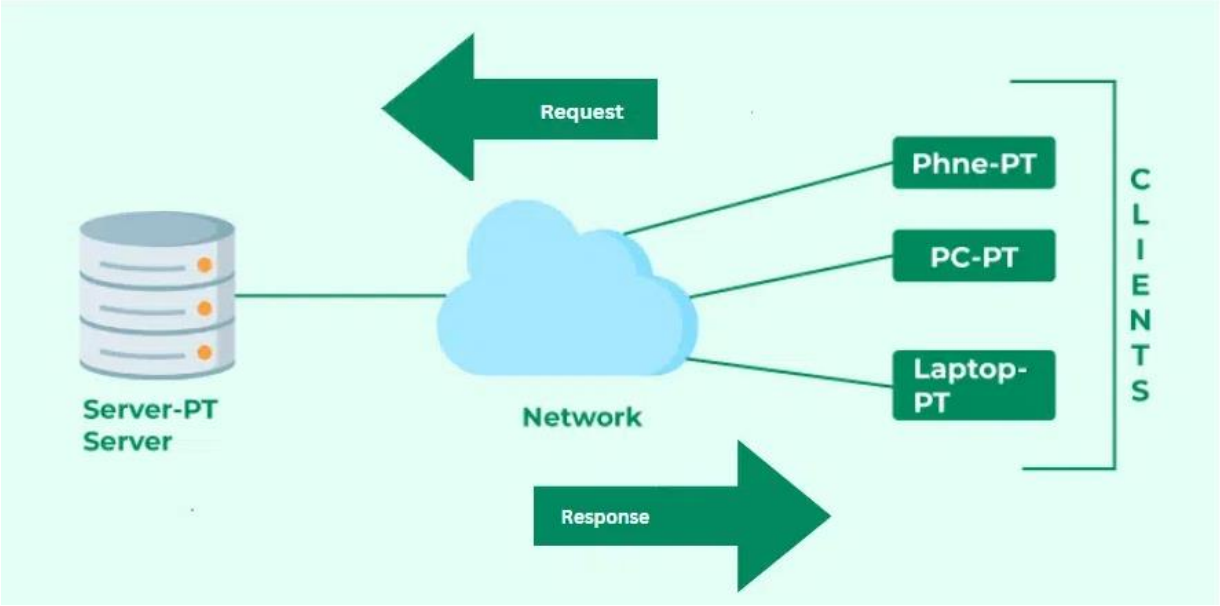


Рисунок 1.1.1 – Графічне зображення моделі «Клієнт-сервер»

Сервер - це потужний комп’ютер або програмне забезпечення, яке розміщує та керує ресурсами, послугами або даними. Він призначений для відповіді на запити клієнтів і надання їм необхідних ресурсів. Сервери зазвичай оптимізовані для високої продуктивності, надійності та масштабованості.

Клієнт — це пристрій користувача (комп’ютер, смартфон, планшет) або програмне забезпечення, яке запитує та споживає ресурси або послуги від сервера. Клієнти спілкуються, надсилаючи запити на сервер і отримуючи відповіді, що містять запитувані дані або послуги.

Клієнти та сервери спілкуються через мережу, яка може бути локальною (LAN) або Інтернетом. Зв’язок може відбуватися за допомогою різних протоколів, таких як HTTP (для веб-додатків), SMTP і IMAP (для електронної пошти), FTP (для передачі файлів) тощо.

У контексті Інтернету веб-сайти є поширеним прикладом моделі клієнт-сервер. Веб-браузери (клієнти) надсилають запити на веб-сторінки веб-серверам, які обробляють ці запити та надсилають назад HTML, CSS та інші ресурси, необхідні для відображення веб-сторінки.

Для взаємодії між клієнтом та сервером у контексті веб-сайтів виконуються наступні кроки:

– Користувач вводить URL (Uniform Resource Locator) веб-сайту або файлу, після чого браузер надсилає запит до DNS-сервера.

– Відбувається пошук адреси WEB-сервера DNS-сервером.

– DNS-сервер надсилає відповідь з IP-адресою Web-сервера.

– Браузер надсилає запити за IP-адресою Web-сервера, використовуючи HTTP/HTTPS.

– Сервер надсилає файли, необхідні для веб-сайту.

– Браузер відтворює файли та відображає веб-сайт. Ця візуалізація виконується за допомогою інтерпретатора DOM (Document Object Model), інтерпретатора CSS і механізму JS, спільно відомого як JIT або компілятор (Just in Time) [4].

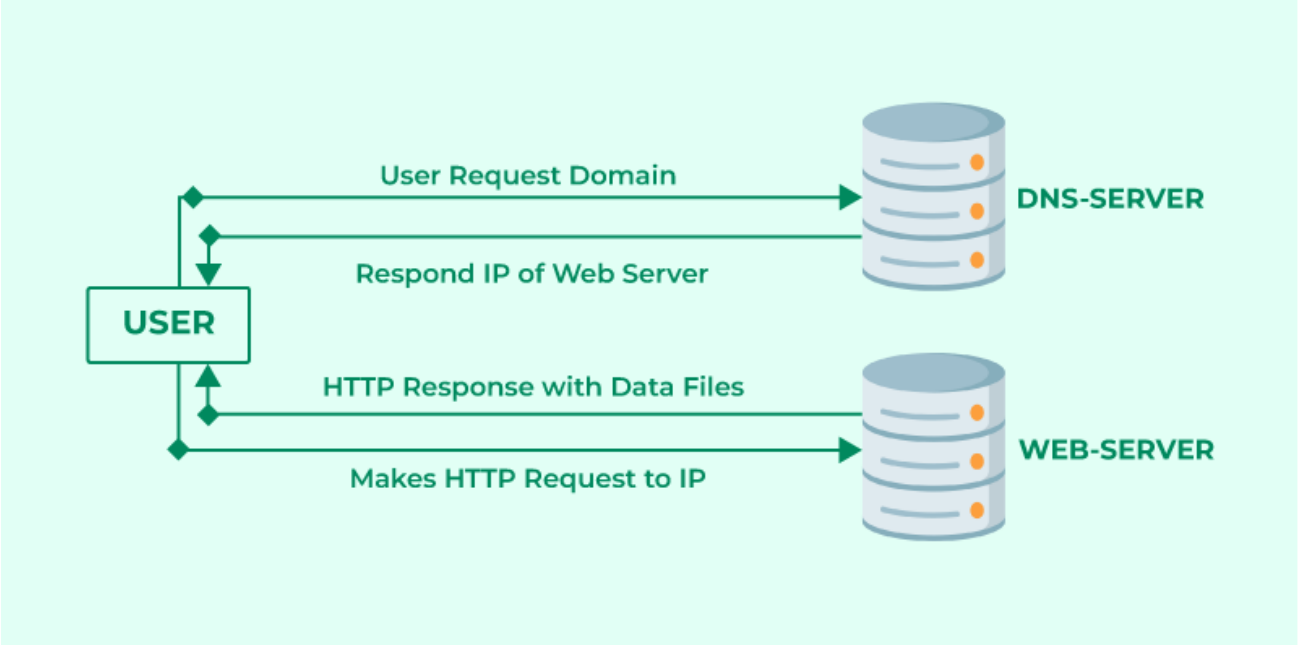


Рисунок 1.1.2 – Графічне зображення взаємодії між клієнтом та сервером у контексті веб-сайтів

Розглянемо переваги та недоліки використання клієнт-серверної моделі.

Переваги:

– Централізована система, яка об'єднує всі дані в одному місці.

– Рентабельність вимагає зниження витрат на обслуговування і можливого відновлення даних.

– Продуктивність клієнта і сервера може бути змінена окремо.

Недоліки:

– Клієнти схильні до зараження вірусами, троянами і хробаками, якщо вони присутні на сервері або завантажені на сервер.

– Сервери схильні до атак типу “Відмова в обслуговуванні (DOS)”.

– Пакети даних можуть бути підроблені або змінені під час передачі.

– Фішинг або захоплення облікових даних для входу чи іншої корисної інформації користувача, а також атаки типу MITM (Людина посередині)

Отже, архітектура клієнт-сервер об’єднує ресурси на серверах для кращого контролю та безпеки, забезпечує гнучкі параметри клієнта та покладається на надійну мережу для масштабованості та ефективності. Незважаючи на фінансові наслідки, модель клієнт-сервер залишається фундаментальною.

1.1.2. Фронтенд та бекенд.

Фронтенд і бекенд є двома ключовими компонентами в архітектурі веб-додатків. Разом вони забезпечують повну функціональність веб-сайту або веб-додатку, взаємодіючи між собою для обробки запитів користувача, зберігання даних і відображення інформації.

Частина веб-сайту, з якою користувач безпосередньо взаємодіє, називається фронтенд. Це включає в себе весь інтерфейс користувача (UI) та досвід користувача (UX). Основні технології, які використовуються для розробки фронтенду, включають HTML, CSS та JavaScript. За допомогою цих технологій розробники створюють структуру, стиль та динамічні елементи сторінок, з якими користувач взаємодіє. Чуйність і продуктивність є двома головними цілями Front End. Розробник повинен переконатися, що сайт адаптивний, тобто він правильно відображається на пристроях будь-якого розміру, жодна частина веб-сайту не повинна працювати ненормально, незалежно від розміру екрана [12].

Розглянемо детальніше мови, за допомогою яких будується фронтенд частина веб-сайту:

– HTML (HyperText Markup Language) є основною мовою розмітки для створення веб-сторінок. Він використовується для визначення структури та змісту вебсторінки за допомогою тегів. Кожен елемент HTML складається з тегів, які обрамляють текст або інші елементи, визначаючи їх функцію на сторінці.

– CSS (Cascading Style Sheets) - це мова стилів, яка використовується для визначення зовнішнього вигляду HTML-елементів. CSS дозволяє змінювати кольори, шрифти, розміри, відступи, розташування і багато інших аспектів візуального представлення веб-сторінки.

– JavaScript - це мова програмування, яка використовується для додавання інтерактивності та динамічних елементів до веб-сторінок. JavaScript може змінювати HTML та CSS на льоту, реагуючи на дії користувача або інші події.

Бекенд - це частина програмного забезпечення або веб-додатку, яка відповідає за обробку даних, логіку та управління базами даних. Він функціонує на сервері і є "невидимою" для кінцевого користувача частиною системи. Основні

завдання бекенду включають обробку запитів від клієнтської сторони (фронтенду), управління базами даних, зберігання, обробку та отримання даних з баз даних, виконання бізнес-логіки додатку, яка визначає, як повинні оброблятися та передаватися дані, аутентифікацію та авторизацію, перевірку прав користувачів та управління доступом до ресурсів, а також інтеграцію з іншими сервісами для отримання або відправки даних. Серверна частина програми, яку іноді називають внутрішньою, управляє загальною функціональністю веб-додатка. Коли користувач взаємодіє з інтерфейсом, він надсилає запит до серверної частини у форматі HTTP. Серверна частина обробляє запит та повертає відповідь [18].

Коли серверна частина обробляє запит, вона зазвичай взаємодіє з такими компонентами:

– серверами баз даних для отримання або зміни відповідних даних;

– мікросервісами, що виконують підмножину завдань, запрошених користувачем;

– API сторонніх розробників для збору додаткової інформації або додаткових функцій.

Налаштування бекенда часто потребує бази даних для керування всією інформацією про клієнтів і продукт. База даних зберігає вміст веб-сайту в структурі, яка полегшує пошук, упорядкування, редагування та збереження даних. Він працює на віддаленому комп'ютері, який називається сервером – саме тому бекенд-розробку іноді називають розробкою на стороні сервера.

Існує багато різних баз даних, які широко використовуються, наприклад:

– MySQL.

– SQL Server.

– PostgreSQL.

– Oracle.

Серверна частина будується за допомогою наступних мов програмування (в тому числі мови, які можуть розпізнати бази даних):

– PHP - це мова сценаріїв на стороні сервера, розроблена спеціально для веб-розробки. Оскільки код PHP виконується на стороні сервера, його називають мовою сценаріїв на стороні сервера.

– Python - для написання структур даних і алгоритмів для створення веб-сайтів. Для розробки веб-додатків використовують бібліотеки та фреймворки Python, такі як Flask, Django та NumPy.

– Java - це потужна, типізована мова програмування, широко використовувана для створення масштабованих і надійних бекенд-систем.

– Ruby - мова програмування з відкритим вихідним кодом. Бекенд-розробники можуть використовувати його для швидкого створення нових програм. Ruby включає все, що потрібно для роботи сайту, тому розробники повного стека також часто використовують цю мову. Ruby особливо добре працює на веб-сайтах із високим трафіком.

– C++ - це універсальна мова програмування, яка активно використовується для конкурентного програмування, а також як мова для серверного програмування.

– Node.js - це середовище виконання з відкритим вихідним кодом і крос-платформне для виконання коду JavaScript поза браузером. NodeJS не є фреймворком і не є мовою програмування. Node.js використовується для створення внутрішніх служб, таких як API, наприклад Web App або Mobile App. Він використовується у розробці великими компаніями, такими як Paypal, Uber, Netflix, Walmart тощо [7].

Розглянемо також фреймворки, що використовуються у бекенд-розробці:

– Express – це фреймворк Nodejs, який використовується для розробки на стороні серверу. Він використовується для створення односторінкових, багатосторінкових і гібридних веб-додатків. З його допомогою можна обробляти кілька різних HTTP-запитів.

– Django - це веб-фреймворк на Python, що дотримується шаблону «модель-шаблон-подання». Він використовується для створення великих і складних веб-додатків. Серед його особливостей — швидкість, безпека та можливість масштабування.

– Ruby on Rails - це платформа на стороні сервера, що дотримується шаблону архітектури модель-подання-контролер. Він надає такі структури за замовчуванням, як веб-сервіси, веб-сторінки та бази даних.

– Laravel - надійна платформа веб-додатків для PHP. Особливістю, яка робить його ідеальним, є повторне використання компонентів різних фреймворків для створення веб-додатку.

Також, при розробці на мові Python може використовуватися фреймворк Flask, вразливості якого можна використати для створення завдань у форматі Capture The Flag. Flask-це фреймворк, який використовується для створення веб-додатків на мові програмування Python за допомогою інструментарію Werkzeug та механізму шаблонів Jinja2. Flask відноситься до категорії мікро-фреймворків, які є мінімальною платформою для веб-додатків, які свідомо надають лише базову функціональність. Це означає, що він легкий і простий у використанні, але в той же час забезпечує достатню функціональність для створення повноцінного веб-додатки. Flask надає інструменти для обробки HTTP-запитів, маршрутизації URL-адрес, управління сеансами, управління шаблонами та інтеграції баз даних.

1.1.3. Веб-сервери та сервери додатків.

Сервер - це центральне сховище, де зберігається інформація та комп'ютерні програми, до яких програміст має доступ у мережі. Веб-сервер і сервер додатків є різновидом сервера, який використовується для доставки сайтів, і тому останній має справу з операціями додатків, що виконуються між користувачами та внутрішніми бізнес-додатками організації.

Веб-сервер-це програмний компонент, який надає статичні дані, такі як зображення, файли та текст, на запити клієнта. Сервер додатків надає бізнес-логіку для обчислення відповіді веб-сервера. Обидва терміни використовуються як взаємозамінні, і сьогодні найпопулярнішим серверним програмним рішенням є гібридні веб-програми.

Веб-сервери створені для того, щоб бути продуктивними, відповідаючи на якомога більше запитів за найкоротший проміжок часу з мінімальним використан-

ням ресурсів. Для цього вони розроблені з урахуванням простоти. За замовчуванням, коли веб-сервер отримує запит, він намагатиметься надати запитуваний файл із файлової системи.

Зрештою, окремі веб-сервери прив'язані до IP-адрес і портів, прослуховуючи запити на загальних портах, таких як TCP/80 для HTTP та TCP/443 для HTTPS. Їм може бути призначено багато унікальних імен, навіть для однієї IP-адреси та порту, що обробляється за допомогою концепції, яка називається віртуальними серверами. Веб-сервери також забезпечують середовище, у якому програми на стороні сервера, такі як PHP, Perl, Python та інші, можуть виконувати та надсилати відповіді [24].

У контексті конфігурації, веб-сервери зазвичай легко налаштувати. Багато корпоративних програм покладаються на веб-сервери, щоб безпосередньо відповідати на запити щодо статичних файлів, на відміну від динамічних файлів.

Сервер додатків розширює функціонал веб-сервера, підтримуючи створення динамічного контенту, виконання програмної логіки та інтеграцію з різноманітними ресурсами. Він забезпечує середовище для виконання коду програми та взаємодії з іншими програмними компонентами, такими як системи обміну повідомленнями та бази даних. Використовуючи бізнес-логіку, сервер додатків перетворює дані більш ефективно, ніж веб-сервер.

Оскільки робочі навантаження сервера додатків складніші, ніж робочі навантаження веб-сервера, вони зазвичай налаштовані таким чином, щоб мати більше ЦП, пам'яті та дискових ресурсів порівняно з веб-серверами. Сервери додатків часто вимагають балансування навантаження, що допомагає розподіляти запити між декількома серверами, розподіляючи поточне навантаження всіх додатків і максимізуючи доступність. Веб-сервери часто розміщуються перед серверами додатків, щоб забезпечити цю функцію балансування навантаження.

Функціональні можливості сервера додатків часто включають функції, яких немає у веб-серверах, наприклад обмін повідомленнями, безпека, об'єднання з'єднань із базами даних, підтримка транзакцій, пакетна обробка, заплановані завдання

та черги завдань. Сервери додатків можуть використовувати стратегії масштабованості, як-от балансування навантаження, і методи відмовостійкості, як-от спільні сеанси, для забезпечення високої доступності. Ці функції часто вмикаються веб-серверами, але вимагають, щоб сервер додатків був налаштований для використання функцій (таких як балансування навантаження), які надає веб-сервер, налаштований як веб-проксі. Це дозволяє серверам додатків обробляти все більшу кількість користувачів одночасно та підтримувати продуктивність, коли трафік зростає.

Більшість веб-серверів підтримують плагіни для мов сценаріїв (наприклад, ASP, JSP, PHP, Perl), які дозволяють веб-серверу створювати динамічний вміст на основі логіки на стороні сервера. Зростаюча кількість серверів додатків не тільки включає в себе можливості веб-сервера, але й використовує HTTP як основний протокол і підтримує інші протоколи (наприклад CGI) для взаємодії з веб-серверами.

Розглянемо популярні веб-сервери та сервери додатків:

– Apache HTTP Server. HTTP-сервер Apache, який часто називають просто «Apache», традиційно вважався веб-сервером, ідеальним для середовищ загального хостингу, завдяки можливості користувачів додавати файли конфігурації для окремих віртуальних серверів і шляхів [20].

– NGINX - це веб-сервер, який також може виступати балансувальником навантаження, зворотним проксі-сервером і http-кешем. Неблокуюча, керована подіями архітектура NGINX робить його швидким і масштабованим, і його часто розгортають у стеках Kubernetes.

– Apache Tomcat - це легкий сервер додатків Java з відкритим вихідним кодом, який служить еталонною реалізацією для сервлетів і сторінок Java Server (JSP).

– Zend Server - це сервер для додатків на основі PHP. Сервер Zend підтримує Apache, NGINX і IIS як відповідні параметри веб-сервера. Він також використовує спеціальний веб-сервер lighttpd для свого інтерфейсу адміністрування.

У контексті розробки вразливих веб-сайтів для навчальних кібербезпекових досліджень, для розгортання краще всього використовувати сервери Apache або Nginx.

1.2. Безпека Web-додатків.

1.2.1. Поширені вразливості у веб-додатках.

Можна виділити 10 найпоширеніших вразливостей у веб-додатках станом на 2021-2023 роки [25]:

– Порушений контроль доступу.

– Розкриття конфіденційних даних.

– Підробка запитів на стороні сервера (SSRF).

– SQL-ін’єкція.

– Міжсайтовий скриптінг (XSS).

– Порушена автентифікація.

– Неправильна конфігурація безпеки.

– Недостатній захист від атак грубою силою.

– Слабкі паролі користувачів.

– Використання компонентів із відомими вразливостями.

Розглянемо декілька з вищеописаних вразливостей детальніше. Контроль доступу забезпечує дотримання політики, щоб користувачі не могли діяти поза межами наданих їм дозволів. Збої зазвичай призводять до несанкціонованого розкриття інформації, модифікації чи знищення всіх даних або виконання бізнес-функцій за межами дозволених користувачеві повноважень.

До поширених вразливостей контролю доступу належать:

– Порушення принципу найменших привілеїв або заборони за замовчуванням, коли доступ має бути наданий лише для певних можливостей, ролей або користувачів, але доступний будь-кому.

– Обхід перевірок контролю доступу шляхом модифікації URL-адреси (підміна параметрів або примусовий перегляд), внутрішнього стану програми або HTML-сторінки, або за допомогою інструменту атаки, що модифікує запити до API.

– Дозвіл на перегляд або редагування чужого облікового запису шляхом надання його унікального ідентифікатора (незахищені прямі посилання на об'єкти)

– Доступ до API з відсутніми елементами управління доступом для POST, PUT і DELETE.

– Підвищення привілеїв. Дія від імені користувача без входу в систему або від імені адміністратора під час входу в систему як користувача.

– Маніпуляції з метаданими, такі як відтворення або підробка токену контролю доступу JSON Web Token (JWT), файлів cookie або прихованих полів для підвищення привілеїв або зловживання недійсністю JWT [12].

– Неправильна конфігурація CORS дозволяє отримати доступ до API з несанкціонованих/ненадійних джерел.

– Примусовий перегляд автентифікованих сторінок від імені неавторизованого користувача або привілейованих сторінок від імені стандартного користувача [15].

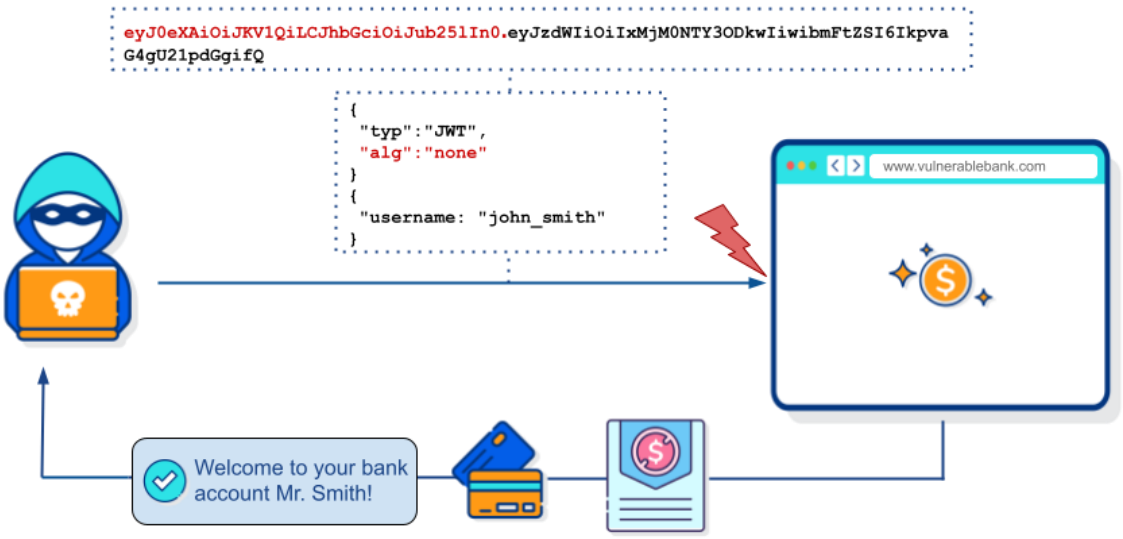


Рисунок 1.2.1 – Графічне зображення атаки з експлуатацією вразливості контролю доступу шляхом зміни алгоритму шифрування JWT-токена

Помилки SSRF виникають щоразу, коли веб-програма отримує віддалений ресурс без перевірки URL-адреси, наданої користувачем. Це дозволяє зловмиснику змусити додаток надіслати підроблений запит до неочікуваного місця призначення, навіть якщо він захищений брандмауером, VPN або іншим типом списків контролю доступу до мережі (ACL). Оскільки сучасні веб-додатки надають кінцевим кори-

стувачам досить зручні функції, отримання URL-адреси стає поширеним сценарієм. Як наслідок, поширеність SSRF зростає. Крім того, серйозність SSRF стає вищою через хмарні сервіси та складність архітектури.

Додаток або веб-сайт є вразливим до різних типів ін’єкцій у випадку якщо:

– Дані, що надаються користувачем, не перевіряються, не фільтруються та не очищуються програмою.

– Динамічні запити або непараметризовані виклики без екранування використовуються безпосередньо в інтерпретаторі.

– Шкідливі дані використовуються безпосередньо. SQL або команда містить структуру і шкідливі дані в динамічних запитах, командах або збережених процедурах.

Деякі з найпоширеніших ін'єкцій - це SQL, NoSQL, команди ОС, об'єктно-реляційне відображення, а також ін'єкції з використанням мови виразів або об'єктно-графічної навігаційної бібліотеки. Також до цієї категорії відносяться ін’єкції, пов’язані з міжсайтовим скриптінгом (XSS).

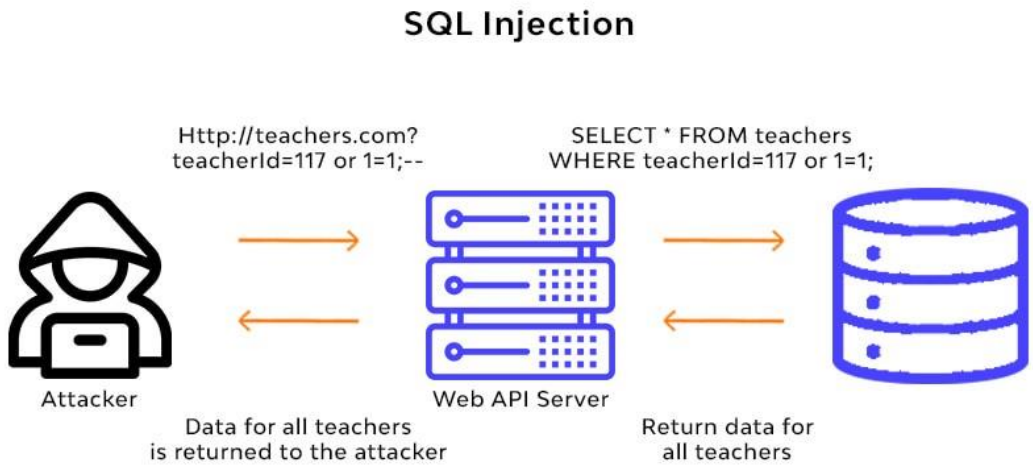


Рисунок 1.2.2 – Графічне зображення атаки з впровадженням ін’єкції SQL

До неправильної конфігурації безпеки відносяться наступні пункти:

– Активовані або встановлені непотрібні функції (наприклад, зайві порти, служби, сторінки, облікові записи чи привілеї).

– Облікові записи за замовчуванням і їхні паролі залишаються активними та незмінними.

– Обробка помилок відображає користувачам надто детальні повідомлення про помилки.

– Сервер не надсилає заголовки або директиви безпеки, або вони не налаштовані на безпечні значення.

– Програмне забезпечення є застарілим або вразливим.

Без узгодженого, повторюваного процесу конфігурації безпеки додатків системи піддаються більшому ризику.

Підтвердження особи користувача, автентифікація та керування сеансами є критично важливими для захисту від атак, пов'язаних з автентифікацією. У вед-додатку можуть бути вразливі місця в автентифікації, якщо він:

– Дозволяє автоматизовані атаки, такі як підміна облікових даних, коли зловмисник має список дійсних імен користувачів і паролів.

– Дозволяє грубу силу або інші автоматизовані атаки.

– Дозволяє використовувати паролі за замовчуванням, слабкі або загальновідомі паролі, такі як «Password1» або «admin/admin».

– Використовує звичайний текст, зашифровані або слабо хешовані сховища даних паролів.

– Не налаштований на багатофакторну автентифікацію.

– Розкриває ідентифікатор сеансу в URL-адресі.

– Неправильно анулює ідентифікатори сеансів, тобто сеанси користувача або токени автентифікації не анулюються належним чином під час виходу з системи або періоду неактивності [15].

Також, швидше за все, веб-сайт буде вразливим у випадках:

– Якщо розробник не знаєте версії всіх використовуваних ним компонентів (як на стороні клієнта, так і на стороні сервера). Це стосується як компонентів, які використовуються безпосередньо, так і вкладених залежностей.

– Якщо програмне забезпечення вразливе, не підтримується або застаріле. Сюди входить операційна система, веб-сервер/сервер додатків, система управління базами даних (СУБД), додатки, API та всі компоненти, середовища виконання та бібліотеки.

– Якщо розробник не виправляє або своєчасно не оновлює базову платформу, фреймворки та залежності з урахуванням ризиків. Це часто трапляється в середовищах, де виправлення є щомісячним або щоквартальним завданням під контролем змін, що залишає організації відкритими для непотрібного впливу виправлених вразливостей на дні або місяці.

– Якщо розробники програмного забезпечення не перевіряють сумісність оновлених, модернізованих або виправлених бібліотек.

Вищеописані вразливості є досить актуальними на даний момент, тому розробка завдань категорії Web у форматі CTF для подальших навчальних кібербезпекових досліджень дозволить розробникам завдань і учасникам змагань детально дослідити дані вразливості як теоретично, так і практично, використовуючи різні техніки зламу систем та їх методів захисту.

1.2.2. Методики захисту веб-додатків.

“Найпотужніша техніка безпеки у веб-розробці — думати про безпеку в усіх, навіть найменших, деталях. Безпека веб-додатків стосується не лише бази коду, а й процесів проекту, зберігання даних, політики компанії тощо. Зрештою, не має значення, наскільки якісним і захищеним є ваш код, якщо ваша документація API є загальнодоступною або пароль суперадміністратора написаний на дошці в офісі.” – Сергій Іванов, Керівник відділу управління ресурами компанії MobiDev [8]. Важко не погодитися з даною цитатою, проте написання безпечного коду, захист баз даних та інші дії, пов’язані з захистом веб-додатків, є критично важливими процесами у ході розробки для максимального зниження ризиків, пов’язаних з атаками на веб-сайти, що використовують випадково залишені вразливості або помилки у конфігурації.

Існують як загальні поради для розробки, так і конкретні дії які необхідно вжити для захисту додатку:

– Перевірка даних: Необхідно переконатися, що всі дані, отримані від користувачів, включно з полями форм і параметрами URL-адреси, перевіряються та очищаються, щоб запобігти таким атакам, як SQL-ін'єкції, міжсайтовий скриптинг (XSS) і командні ін'єкції [3].

– Аутентифікація та авторизація: Впровадження надійних механізмів автентифікації, таких як багатофакторна автентифікація, а також встановлення контролю доступу, який гарантує, що користувачі можуть отримати доступ лише до тих даних і функцій, які їм потрібні.

– Безпечні комунікації: Використовування протоколів HTTPS і SSL/TLS для шифрування даних, що передаються між сервером і клієнтом для запобігання підслуховуванню і підробці даних.

– Шифрування даних: Конфіденційні дані, такі як паролі та інформація про кредитні картки, повинні бути зашифровані за допомогою стійких алгоритмів шифрування, щоб запобігти несанкціонованому доступу.

– Обробка та реєстрація помилок: Впровадження належних механізмів обробки та реєстрації помилок для виявлення вразливостей безпеки та належного реагування на інциденти безпеки.

– Регулярне оновлення безпеки веб-застосунку: Оновлення веб-додату та його базових компонентів найновішими патчами та оновлення безпеки для усунення відомих вразливостей.

– Найменші привілеї: Користувачам і процесам необхідно надавати лише мінімально необхідні права для виконання їхніх завдань, щоб зменшити ризик атак на підвищення привілеїв

– Глибокий захист: Впровадження кількох рівнів контролю безпеки, таких як брандмауери, системи виявлення вторгнень та антивірусне програмне забезпечення для зменшення ризиків виникнення єдиної точки відмови.

Також, під час розробки необхідно враховувати управління сеансами користувачів, що є критично важливим аспектом безпеки веб-додатків, який передбачає керування та контроль сеансів користувачів для запобігання несанкціонованому доступу. Викрадення сеансу та фіксація сеансу є двома поширеними атаками, які можуть скомпрометувати сеанси користувача. Викрадення сеансу відбувається, коли зловмисник викрадає ідентифікатор сеансу користувача (унікальний ідентифікатор, призначений сеансу користувача), що дозволяє йому взяти під контроль сеанс і отримати доступ до конфіденційних даних або виконувати несанкціоновані

дії. Фіксація сеансу відбувається, коли зловмисник встановлює ідентифікатор сеансу користувача на заздалегідь визначене значення, що дозволяє йому взяти під контроль сеанс, коли користувач входить [1].

Практичні поради для керування сеансом користувача:

– Ідентифікатор сеансу має бути досить довгим і складним, щоб зловмисникам було важко вгадати або застосувати грубу силу. Використання випадково згенерованих ідентифікаторів сеансу може допомогти запобігти захопленню сеансу та атакам фіксації.

– Ідентифікатори сеансу слід генерувати повторно після входу користувача або виконання будь-яких конфіденційних дій. Це гарантує, що ідентифікатор сеансу часто змінюється, що ускладнює зловмисникам захоплення або фіксацію сеансу.

– Сеанси повинні мати час закінчення, після якого користувач автоматично виходить із системи. Це допомагає запобігти використанню зловмисниками ідентифікатора сеансу, який був активним протягом тривалого часу.

– Впровадження двофакторної автентифікації може додати додатковий рівень безпеки для сеансів користувачів, ускладнюючи зловмисникам захоплення або фіксацію сеансу.

– Регулярний моніторинг активності користувача може допомогти виявити будь-яку незвичайну поведінку, яка може свідчити про викрадення сеансу або атаку фіксації. Будь-яка підозріла діяльність має бути негайно розслідувана та вжито відповідних заходів.

Якщо веб-сайт надає можливість користувачам завантажувати файли на сервер – це може становити значну загрозу безпеці. Зловмисники можуть завантажувати шкідливі файли, які поставлять під загрозу безпеку всієї програми. Необхідно застосовувати безпечні механізми завантаження файлів, щоб забезпечити можливість завантаження лише авторизованих файлів.

Аналіз вихідного коду є найкращим методом виявлення вразливості додатків до ін'єкцій та ряду інших вразливостей. В ході розробки наполегливо рекомендується автоматизоване тестування всіх параметрів, заголовків, URL, файлів cookie та вхідних даних.

У контексті змагань, де за мету ставиться захоплення прапорів, Web-категорія часто може бути пов’язаною з напрямком тестування на проникнення (Penetration testing), що є однією з важливих методик захисту веб-додатків.

Тестування на проникнення – це контрольований та систематичний процес перевірки безпеки системи або додатку шляхом імітації атак з боку зловмисників, метою якого є виявлення вразливостей, які можуть бути використані для несанкціонованого доступу, викрадення даних, або інших зловмисних дій. Пентестери (особи, що виконують тестування на проникнення) діють як справжні загрозливі особи - використовують вразливості, отримують несанкціонований доступ, викрадають дані та порушують роботу служб. Однак вони роблять це за контрактом із власником веб-додатку в узгодженому обсязі та без завдання реальної шкоди організації. Цей метод тестування є більш складним для виконання, але він може виявити додаткові ризики, які автоматизовані інструменти можуть пропустити.

Звичайний процес тестування на проникнення для веб-додатків включає в себе сканер вразливостей, який використовується для дослідження та виявлення прогалин у системі безпеки, таких як неправильна конфігурація, не виправлене програмне забезпечення, SQLi, міжсайтовий скриптинг тощо. Після виконання тестування сканером вразливостей, пентестери, використовуючи ручний спосіб, входять у систему для того, щоб підтвердити справжність вразливостей, знайдених сканером та пошуку більш складних вразливостей.

Види тестування на проникнення поділяються на:

– Black Box.

– White Box.

– Gray Box.

Black Box – це вид тестування, при якому тестери не мають жодної попередньої інформації про систему, яку вони тестують, що імітує реальний сценарій дій зовнішніх зловмисників.

White Box – це тестування, при якому тестери мають повну інформацію про цільову систему, включаючи вихідний код, архітектуру та конфігурації. Це дозволяє більше детально і глибоко перевірити безпеку системи.

Gray Box – це компромісний підхід, коли тестери мають часткову інформацію про систему, що дозволяє зосередитися на найбільш критичних аспектах безпеки.

Під час змагань у форматі CTF за бажанням організаторів для імітації одного з трьох вищеописаних пунктів учасникам може бути доступним вихідний код вебдодатків, який необхідно проаналізувати, виявити та використати вразливості для зламу веб-сайтів/систем та захоплення прапорів.

1.2.3. Інструменти для тестування безпеки веб-застосунків.

Варто розпочати з операційної системи, яка найкраще підійде для тестування на проникнення. Kali Linux (раніше відомий як BackTrack Linux) - це дистрибутив Linux з відкритим вихідним кодом, заснований на Debian, призначений для розширеного тестування на проникнення та аудиту безпеки [26]. Це досягається шляхом надання загальних інструментів, конфігурацій та автоматизації, що дозволяє користувачеві зосередитися на завданні, яке потрібно виконати, а не на супутніх діях.

Kali Linux містить галузеві модифікації, а також кілька сотень інструментів, орієнтованих на різні завдання інформаційної безпеки, такі як тестування на проникнення, дослідження безпеки, комп'ютерна криміналістика, зворотний інжиніринг, управління вразливостями та тестування червоних команд. Kali Linux є багатоплатформним рішенням, доступним і вільно розповсюджуваним серед професіоналів та аматорів інформаційної безпеки.

Дана операційна система загалом надає понад 600 інструментів для тестування на проникнення. У порівнянні з дистрибутивом BackTrack Linux, розробники Kali Linux виключили велику кількість інструментів, які або просто не працювали,

або дублювали інші інструменти, які надавали таку ж або подібну функціональність. Деякими найвідомішими інструментами, які включені в даний дистрибутив є Wireshark, Metasploit, Nmap, John the Ripper та інші.

Також Kali Linux відомий широким спектром інструментів експлуатації вразливостей, серед яких є автоматичні ін’єкції в SQL-запити, тестування безпеки хмарних служб та сканери вразливостей.

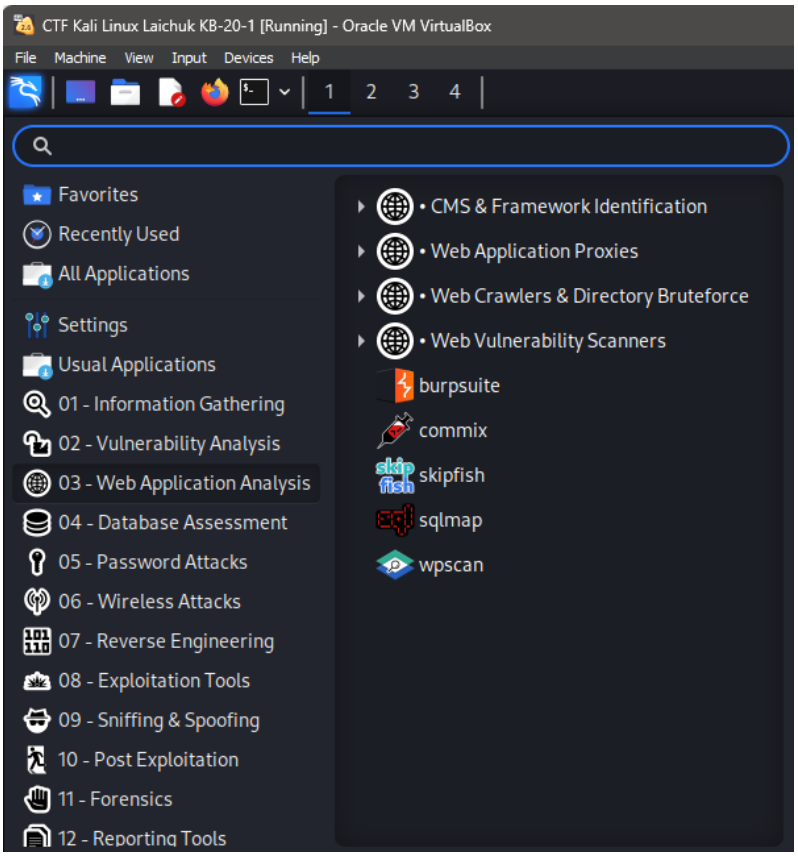


Рисунок 1.2.3 – Перелік усіх категорій з інструментами для тестування на проникнення, що надає дистрибутив Kali Linux

Першим у черзі інструментів для тестування безпеки є Burp Suite. Burp Suite - це комплексний інструмент для тестування безпеки веб-застосунків, розроблений компанією PortSwigger. Основним функціоналом Burp Suite є перехоплення і модифікація HTTP/HTTPS-запитів між браузером і сервером, що дозволяє детально аналізувати і змінювати трафік для виявлення вразливостей. Наприклад, можна вручну змінювати параметри запитів для перевірки наявності SQL-ін'єкцій, XSS-атак або інших видів вразливостей.

Крім цього, Burp Suite пропонує автоматизовані інструменти для сканування веб-додатків, які допомагають швидко знаходити відомі вразливості. Ці інструменти можуть аналізувати структуру веб-застосунку, виявляти його сторінки і форми, перевіряти вхідні дані на наявність уразливостей.

Найбільше ефективним є використання Burp Suite у поєднанні з іншими інструментами, такими як Nmap, Wireshark та Metasploit.

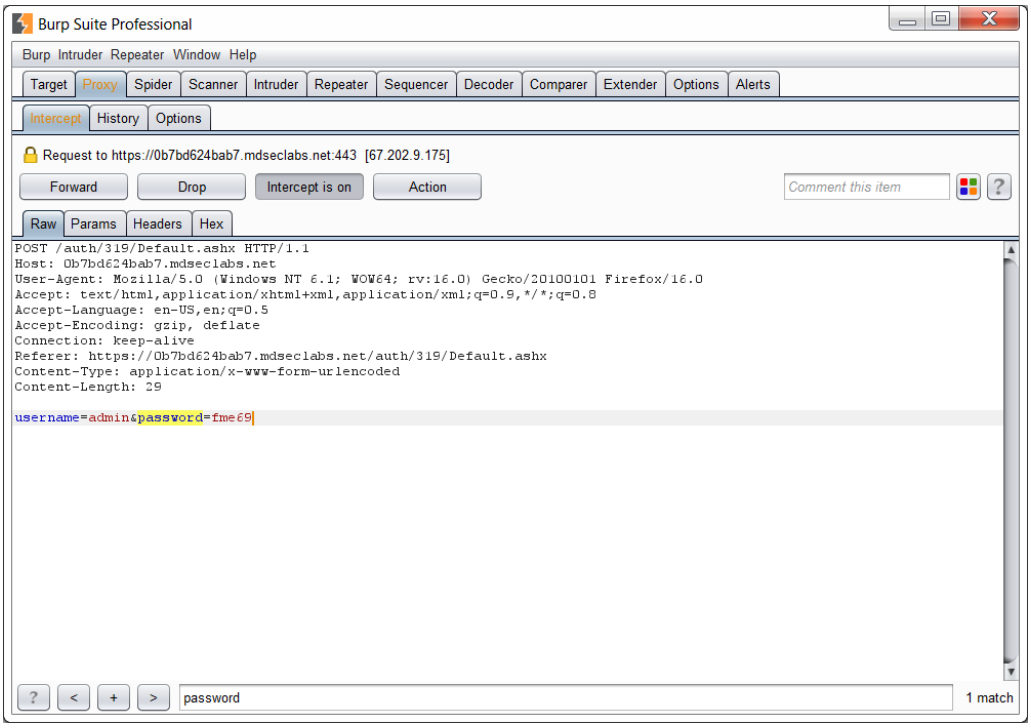


Рисунок 1.2.4 – Інтерфейс інструмента для тестування веб-додатків - Burp Suite

Metasploit — це фреймворк, який можуть використовувати як зловмисники, так і етичні хакери для виявлення системних вразливостей у мережах і серверах. Завдяки відкритому вихідному коду, його легко налаштувати та використовувати з більшістю операційних систем.

За допомогою Metasploit тестувальники можуть використовувати готовий або створений код і впроваджувати його в мережу для пошуку слабких місць. Metasploit містить велику базу даних відомих експлойтів і дозволяє користувачам створювати власні модулі для специфічних сценаріїв.

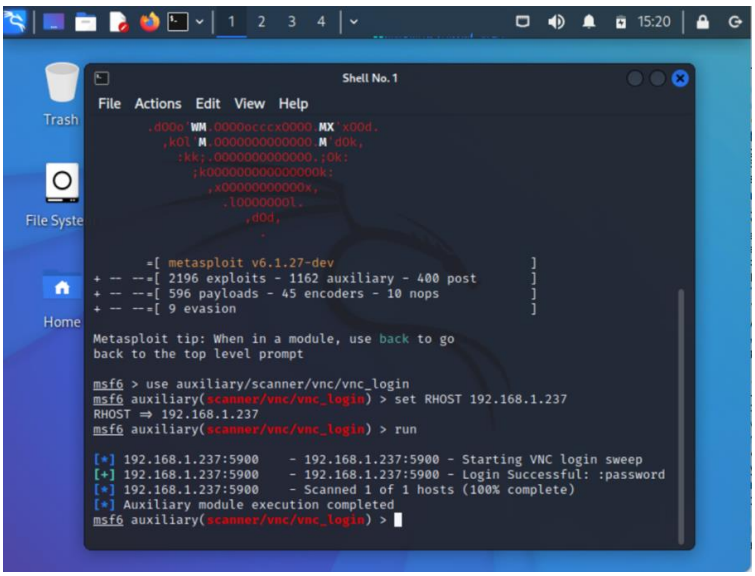


Рисунок 1.2.5 – Приклад проведення атаки за допомогою фреймворка Metasploit та операційної системи Kali Linux

Nikto - це програмне забезпечення з відкритим вихідним кодом, написане на мові Perl, яке використовується для сканування веб-сервера на наявність вразливостей, які можуть бути використані і скомпрометувати сервер. Він також може перевіряти застарілі версії 1200 серверів і може виявляти проблеми з певними версіями більш ніж 200 серверів [27]. Він також може створювати відбитки сервера за допомогою файлів favicon.ico, присутніх на сервері. Він не призначений для того, щоб бути особливо непомітним інструментом, а скоріше для того, щоб бути швидким і ефективним для виконання завдання за дуже короткий час. Завдяки цьому веб-адміністратор може легко виявити, що його сервер сканується, переглянувши файли журналів.

Він також може виявити деякі елементи, які не мають проблем з безпекою, а є лише інформацією, яка показує, як її використовувати, щоб захистити веб-сервер більш належним чином.

Можливості інструмента Nikto:

– Повна підтримка SSL.

– Знаходить субдомен.

– Підтримує повний HTTP проксі.

– Звіт про застарілі компоненти.

– Вгадування імені користувача.

– Надає детальну інформацію про встановлене програмне забезпечення.

– Приймає файл Nmap як вхідні дані для сканування порту на веб-сервері.

– Вміє виконувати словникову атаку.

– Легко оновлюється.

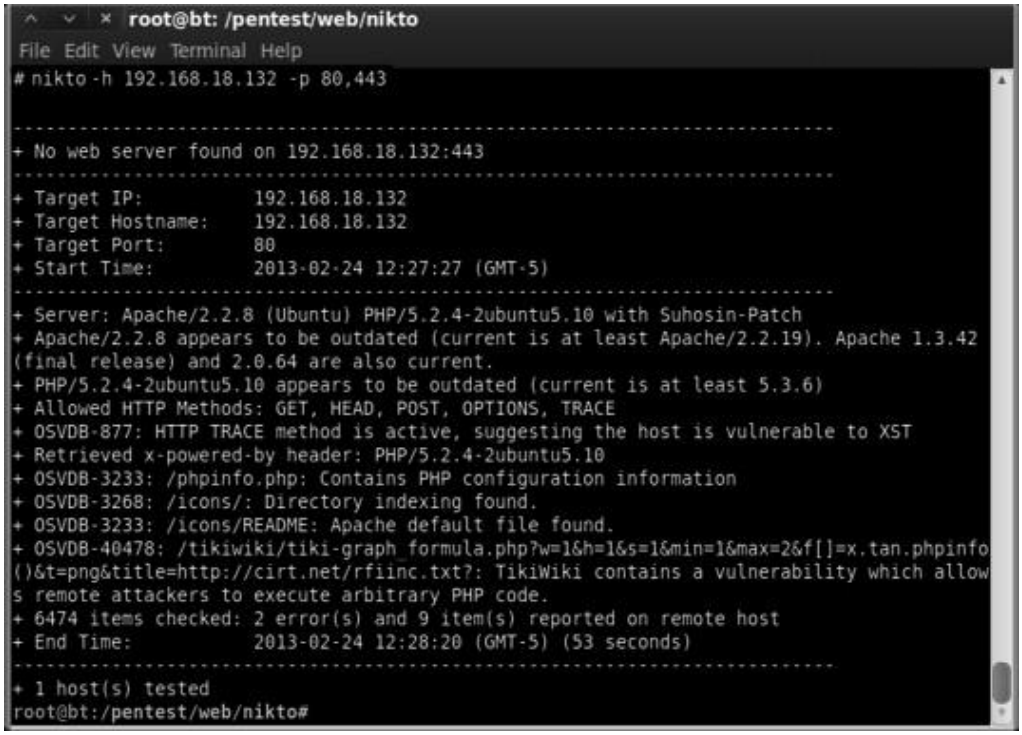


Рисунок 1.2.6 – Приклад сканування веб-сайту за допомогою інструмента Nikto

Також, у разі необхідності проведення атак на паролі, що зберігаються у вигляді хешів, можна використати такі інструменти як John the Ripper або Hashcat. Hashcat – це один із найпотужніших інструментів для відновлення паролів, відомий своєю ефективністю та швидкістю, який використовується фахівцями з кібербезпеки для перевірки стійкості паролів і виявлення слабких місць у системах аутентифікації. Hashcat підтримує широкий спектр алгоритмів хешування, таких як MD5, SHA-1, SHA-256, NTLM, bcrypt та багато інших. Це дозволяє використовувати його для відновлення паролів у різних системах та застосунках. Однією з головних переваг Hashcat є його здатність використовувати потужність графічних процесорів (GPU), що значно прискорює процес обчислення хешів у порівнянні з традиційними методами на основі центрального процесора (CPU). Основні методи атаки, які підтримує Hashcat, включають брутфорс-атаки, атаки за словником, комбінаційні атаки, маскові атаки та гібридні атаки.

John the Ripper, з іншого боку, спочатку був розроблений як інструмент для зламу паролів Unix, але з часом його функціональність значно розширилася. Він підтримує численні алгоритми хешування і може працювати на різних платформах. Обидва інструменти підтримують різні методи атак, включаючи брутфорс, атаки за словником, маскові атаки та інші. Втім, Hashcat має перевагу в гнучкості та масштабованості, завдяки своїй здатності працювати з великими обчислювальними потужностями, такими як розподілені системи та кластерні обчислення.

```
beb4de0dd3955303cf50fe548006b7c5:alligator2023

Session..........: hashcat
Status...........: Cracked
Hash.Mode........: 1000 (NTLM)
Hash.Target......: beb4de0dd3955303cf50fe548006b7c5
Time.Started.....: Wed Jun 14 11:43:30 2023 (0 secs)
Time.Estimated...: Wed Jun 14 11:43:30 2023 (0 secs)
Kernel.Feature...: Pure Kernel
Guess.Base.......: File (animals.txt), Left Side
Guess.Mod........: Mask (?d?d?d?d) [4], Right Side
Guess.Queue.Base.: 1/1 (100.00%)
Guess.Queue.Mod..: 1/1 (100.00%)
Speed.#1.........:  2117.0 kH/s (0.24ms) @ Accel:64 Loops:256 Thr:128 Vec:1
Recovered........: 1/1 (100.00%) Digests (total), 1/1 (100.00%) Digests (new)
Progress.........: 1024/40000 (2.56%)
Rejected.........: 0/1024 (0.00%)
Restore.Point....: 0/4 (0.00%)
Restore.Sub.#1...: Salt:0 Amplifier:0-256 Iteration:0-256
Candidate.Engine.: Device Generator
Candidates.#1....: alligator1234 -> donkey4323
Hardware.Mon.#1..: Temp: 51c Fan:  0% Util: 65% Core:2520MHz Mem:10501MHz Bus:16
```

Рисунок 1.2.7 – Приклад успішної атаки за словником на хеш за допомогою інструмента Hashcat

Також, чимало інструментів для тих чи інших завдань щодо тестування на проникнення знаходяться у відкритих джерелах та репозиторіях, таких як GitHub тощо.

1.3. Розробка завдань Web-категорії для CTF-змагань.

1.3.1. Рівні складності завдань категорії Web у форматі захоплення прапора.

У сфері кібербезпеки змагання «Захоплення прапора» (Capture the Flag, CTF) - це вправи, в яких учасники, індивідуально або в складі команди, повинні знайти і використати вразливості в системі, щоб захопити «прапор» або частину інформації.

При розробці завдань у форматі захоплення прапорів варто розподіляти їх на певні рівні відповідно до складності їх вирішення, щоб вміст конкретного завдання

не перевищував цільової складності. Категорія веб-безпеки часто містить спеціально розроблені веб-програми, які містять деякі недоліки веб-безпеки, які необхідно виявити та використати.

Загалом, складність завдань поділяють на 6 рівнів, відлік яких починається з 0. Рівень також напряму повинен залежати від кількості дій, які учасник повинен виконати для знаходження вразливості та захоплення прапора, тому розглянемо кожен рівень детальніше.

Завдання рівня 0 – це завдання, що не потребують особливих технічних знань. На даному рівні майже у всіх випадках рішення можна знайти за допомогою пошуку інформації в пошукових системах, таких як Google, або простих маніпуляцій. Для вирішення завдань рівня 0 не обов'язково мати термінал, віртуальну машину чи певну операційну систему. Очікується, що більшість учасників вирішить це завдання, тому при розробці необхідно залишати досить зрозумілі та очевидні підказки. Знання чи досвід програмування не потрібні, адже завдання вважається вступним. В контексті Web-категорії завдання рівня 0 – це завдання, вирішити які можна переглянувши вихідний код (HTML, CSS або JavaScript), в якому буде знаходитись прапор або його частини, тому вони оцінюються мінімальною кількістю балів.

Завдання рівня 1 не вимагають жодних знань про вразливі місця чи техніку, але для їх вирішення може знадобитися спеціальний інструмент. Такі завдання можуть вимагати базових попередніх знань. На цьому рівні фундаментальні дослідження повинні привести до базового використання інструменту або концепції. Також для захоплення прапору може знадобитися термінал. Має місце можливість запускати віртуальну машину або інший спосіб запуску лише інструмента CLI, але слід докладати зусиль, щоб цього уникнути. На рівні 1 в завданнях категорії Web – це завдання, в якому учаснику необхідно видобути спеціальний "cookie" або секретне значення з вихідного коду сайту (або дії, що відповідають описаним за кількістю маніпуляцій та їх складністю). Завдання даного рівня оцінюються кількістю балів, що є нижче середнього значення.

Випробування рівня 2 вимагають простого знання однієї вразливості. Вирішення таких завдань можуть потребувати достатніх знань для проведення онлайн-дослідження та знаходження рішення в Інтернеті з подальшою адаптацією під відповідну вразливість. На даному рівні знання програмування не є обов'язковими, але можуть бути дуже корисними. В контексті категорії Web завдання рівня 2 можуть включати вразливості впровадження команд (Command Injection). Успішне захоплення прапора оцінюється середньою кількістю балів від максимального.

Завдання рівня 3 та 4 в категорії Web досить схожі між собою, відрізняючись лише складністю експлуатації схожих вразливостей. Загалом, на рівні 3 потребуються реальні знання про групи вразливостей. Учасник не зможе просто скопіювати і використати онлайн-рішення, не обійшовши інші механізми захисту. На даному рівні розробники можуть залишати підказки, які скерують учасників на правильний крок. Настійно рекомендується знання програмування, оскільки на цьому етапі очікується, що потрібні спеціальні інструменти, сценарії або модифікації існуючих інструментів.

Завдання рівня 4 вимагають глибоких знань про певний тип уразливості чи концепції та часто включають кілька наборів уразливостей, де використання однієї вразливості призведе до області, де потрібно використовувати іншу. Зазвичай рівні 3 та 4 містять такі вразливості як SQLi (SQL Injection), XSS (Cross-Site Scripting), CSRF (Cross-Site Request Forgery). Захоплення прапорів на такому рівні оцінюється в кількість балів, що є вищою за середнє значення.

Завдання рівня 5 вимагають глибокого розуміння певної вразливості або структури. Іноді вони використовують невідомі вразливості в програмах або CVE, а іноді вони вимагають глибокого читання таких речей, як посібники або документації. Хоча людина може глибоко знати певну концепцію, вона може не знати про нові ідеї в цій галузі. Також розробниками можуть залишатися ті чи інші підказки, і в той же час, учасникам потрібні глибокі знання програмування (не виключається використання готових шаблонів-експлойтів для експлуатації вразливостей). У Web-категорії до рівня 5 відносяться CVE або помилки у фреймворках

веб-додатків, які необхідно дослідити, знайти вразливе місце для входу та використати для успішного захоплення прапора. Завдання даного рівня оцінюються у максимальну кількість балів.

1.3.2. Вибір категорії та створення завдань.

Для початку необхідно проаналізувати та здійснити вибір підкатегорій для їх подальшої реалізації у вигляді вразливих місць на веб-сайті. Це залежить від того, яка мета та цілі ставляться перед розробкою завдань та перед учасниками: дослідження певної однієї вразливості, здобуття теоретичних та практичних навичок її експлуатації, чи дослідження групи вразливостей та механік отримання доступу до файлів або загалом до веб-сервера, поєднавши вразливості один з одним. До таких підкатегорій відносяться вразливості, пов'язані з впровадженням ін'єкцій в SQL-запити (SQLi), вразливості, які дозволяють впроваджувати шкідливі скрипти (XSS), вразливості, що дозволяють виконувати шкідливий код на сервері (RCE), вразливості пов'язані з обходом шляхів (Directory/Path Traversal), включенням небезпечних файлів (File Inclusion) та інші на розсуд організаторів та розробників завдань (ін'єкції різних типів, вразливості у версіях інструментів, бібліотек, фреймворків і т.д.).

Процес розробки завдань складається з наступних пунктів:

– Вибір теми завдання.

– Розробка сценарію атаки.

– Підготовка оточення, розробка вразливого веб-сайту.

Першим кроком у створенні завдання у форматі CTF є вибір теми. Виконуючи цей пункт варто враховувати, що тема завдання повинна бути такою, щоб учасники могли зрозуміти її контекст і знайти вразливості. Прикладами тем можуть бути вразливий блог або форум, де користувачі можуть створювати, редагувати та коментувати записи, онлайн-магазин, який дозволяє користувачам переглядати товари, додавати їх до кошика та здійснювати покупки, система управління вмістом (CMS) тощо.

Після вибору теми завдання необхідно розробити сценарій атаки з описом експлуатації вразливостей учасниками для успішного захоплення прапорів. Наприклад, розроблений веб-сайт надає учасникам можливість завантажувати зображення на сервер з їх подальшим переглядом за посиланням, проте формат файлів або їх вміст не перевіряється належним чином на шкідливий вміст. В такому випадку, за сценарієм користувач завантажує PHP-файл замість зображення, що містить шкідливий код для віддаленого виконання команд на сервері (RCE) або отримання зворотньої оболонки від сервера до робочої станцію атакуючого (Reverse Shell). Після цього, отримавши доступ до файлів на сервері та використовуючи необхідні команди для їх знаходження та перегляду, учасник успішно знаходить прапор.

Третій пункт у процесі розробки завдань містить в собі підготовку оточення та розробку вразливого веб-сайту. На даному етапі, в залежності від обраної мови програмування, фреймворків тощо, потрібно встановити необхідні залежності, бібліотеки, створити та налаштувати базу даних (за необхідності), встановити та налаштувати веб-сервер, підготувати операційну систему шляхом створення файлів та каталогів, необхідних для функціонування веб-сайту, а також створення користувача з обмеженими правами (за необхідності, адже використовуючи веб-сервери Apache або Nginx, за замовчуванням користувачем буде www-data). Створення веб-сайту шляхом написання коду попередньо використавши вразливі функції є фінальним етапом саме процесу розробки завдання, після чого необхідно перейти до пункту тестування та валідації завдань.

1.3.3. Тестування та валідація завдань.

Тестування завдань включає в собі наступні пункти:

– Перевірка функціональності.

– Виявлення вразливостей.

– Перевірка сценаріїв атак.

– Оцінка складності.

Для початку необхідно упевнитися, що завдання працює коректно і відповідно до опису – перевіряється весь основний функціонал, не враховуючи вразли-

вості. Після цього потрібно виконати цілеспрямоване застосування атак згідно сценаріїв, описаних при розробці завдання, щоб переконатися, що учасник дійсно зможе успішно експлуатувати вразливості та успішно захопити прапори. В залежності від дозволених методів експлуатації можна використовувати як ручне тестування, так і автоматизоване, використовуючи інструменти для тестування безпеки, такі як Burp Suite, OWASP ZAP, Nikto тощо, а також скрипти для автоматизації тестування.

Для успішної валідації завдань потрібно переконатися, що завдання відповідають технічним вимогам та специфікаціям, а рівень їх складності відповідає очікуванням для даної категорії. Також важливо впевнитися, що існує чіткий та логічний спосіб вирішення завдань.

У випадку виявлення проблем необхідно виправити помилки у коді/конфігураційних файлах для коректної роботи веб-сервера та веб-сайту, за необхідності провести оптимізацію складності, після чого виконати остаточну перевірку.

Таким чином, процес тестування та валідації завдань є критичним для забезпечення їх якості, відповідності стандартам CTF-змагань, а також навчальної цінності для учасників.

РОЗДІЛ 2. ПРОЕКТУВАННЯ ВЕБ-ДОДАТКІВ ТА НАБОРУ ЗАВДАНЬ

2.1. Аналіз вимог до набору завдань.

2.1.1. Цільова аудиторія та контекст застосування.

Цільова аудиторія завдань, що розробляються, складається з початківців у сфері тестування на проникнення та студентів університетів. До першої групи належать особи, які тільки починають вивчати основи тестування на проникнення та мають базові знання у сфері кібербезпеки. Ці люди можуть мати загальне уявлення про мережеві протоколи, операційні системи та основні принципи програмування, але їхні навички в сфері тестування на проникнення ще знаходяться на початковому рівні. Завдання для цієї аудиторії мають бути розроблені таким чином, щоб поступово поглиблювати їхні знання та розвивати практичні навички в реальних сценаріях кібербезпеки.

Друга група цільової аудиторії складається зі студентів, які навчаються на спеціальностях, пов'язаних з інформаційною безпекою. Ці студенти вже мають певний рівень теоретичних знань, отриманих під час навчання, і прагнуть застосувати ці знання на практиці. Завдання для студентів повинні бути інтегровані в навчальний процес таким чином, щоб доповнювати теоретичні лекції практичними вправами, які допоможуть їм краще зрозуміти та закріпити вивчений матеріал.

Обидві групи мають спільну мету - підвищення своїх знань та навичок у сфері кібербезпеки, а завдання мають стати ефективним інструментом для досягнення цієї мети.

Виходячи із вищеописаної інформації, обидві групи на момент виконання завдань повинні мати наступні попередні знання та навички:

– Розуміння принципів роботи мереж, мережевих протоколів та основних мережевих концепцій.

– Базові знання будь-якої мови програмування (наприклад, Python або JavaScript), які дозволять виконувати прості скрипти та розуміти логіку програм.

– Розуміння роботи операційних систем, а особливо Unix-подібних систем.

– Основні концепції та принципи кібербезпеки, включаючи різні види атак та методи захисту.

2.1.2. Навчальні цілі.

Навчальні цілі при розробці завдань для змагань CTF з категорії Web загалом спрямовані на поглиблення розуміння учасників у різних аспектах веб-безпеки, розвитку навичок виявлення та експлуатації вразливостей, а також покращення практичних умінь у створенні безпечних веб-додатків [21].

Однією із головних цілей розробки завдань – вивчення інструментарію, що надається операційною системою Kali Linux для проведення тестування безпеки, виконання атак на окремі компоненти веб-сайтів та загальний огляд певних ситуацій, які часто зустрічаються під час змагань формату Capture The Flag.

У даній кваліфікаційній роботі для створення завдань будуть використані деякі з найпопулярніших вразливостей веб-додатків, описаних у п. 1.2.1 та інші, а саме:

– Порушена аутентифікація (використання слабкого секретного ключа в токені JSON Web (JWT) з подальшою підміною властивостей в тілі токена).

– SQL-ін’єкція.

– Cross-Site Scripting (XSS).

– Слабкі паролі користувачів.

– Використання компонентів із відомими вразливостями.

– Server-Side Template Injection (SSTI).

Також, окрім вразливостей, буде застосований автоматичний пошук прихованих каталогів та файлів на веб-сервері, перехоплення та модифікація HTTP-запитів перед їх відправкою до веб-сервера.

Завдання, яке пов’язане з порушеною аутентифікацією через використання слабкого секретного ключа в JWT-токені, полягає в тому, щоб навчити учасників розпізнавати та експлуатувати вразливості, пов’язані з неправильною конфігурацією та використанням токенів для аутентифікації та авторизації. Ціль полягає в тому, щоб зрозуміти, як важливо забезпечити належний захист секретних ключів, які використовуються для підпису токенів.

Навчальна ціль завдання з SQL-ін’єкцією полягає в ознайомленні учасників з вразливостями, які виникають через неналежну обробку SQL-запитів. Учасники

мають навчитися виявляти та використовувати такі вразливості, вставляючи шкідливий SQL-код у запити або використовуючи автоматичні інструменти для генерування SQL-ін’єкцій. Також, після проведення атак, учасники повинні зрозуміти методи захисту, такі як параметризовані запити або екранування (санітизація), для запобігання SQL-ін’єкціям.

Мета завдання з вразливістю XSS - навчити учасників розпізнавати та експлуатувати вразливості, що виникають через неналежну валідацію або екранування введених даних.

Для слабких паролів користувачів мета - навчити учасників розпізнавати і експлуатувати слабкі паролі, демонструючи важливість використання складних і унікальних паролів. Учасники можуть навчитися методам брутфорсінгу та атакам словникового типу, а також зрозуміти, як впроваджувати політики щодо безпечних паролів у веб-додатках.

Щодо використання компонентів із відомими вразливостями, мета полягає у навчанні учасників знаходити та ідентифікувати застарілі або вразливі інструменти, бібліотеки та фреймворки, які використовуються в додатках. Учасники навчаться розуміти важливість регулярних оновлень компонентів.

Server-Side Template Injection (SSTI) має на меті навчити учасників розпізнавати та експлуатувати вразливості, пов'язані з обробкою шаблонів на стороні сервера. Учасники зрозуміють, як шаблони можуть бути вразливими до ін'єкцій, як експлуатувати такі вразливості для виконання довільного коду на сервері та які заходи безпеки можна застосовувати для захисту від SSTI. Ці навчальні цілі спрямовані на підвищення обізнаності та компетентності учасників у сфері веб-безпеки, надаючи їм практичні навички виявлення та виправлення різних типів вразливостей у веб-додатках.

2.2. Розробка специфікацій завдань.

2.2.1. Проектування та опис завдань.

Спроектуємо завдання та розглянемо описи до кожного з них, виділяючи основний функціонал та вразливості на веб-сайтах з подальшим визначенням їх складності.

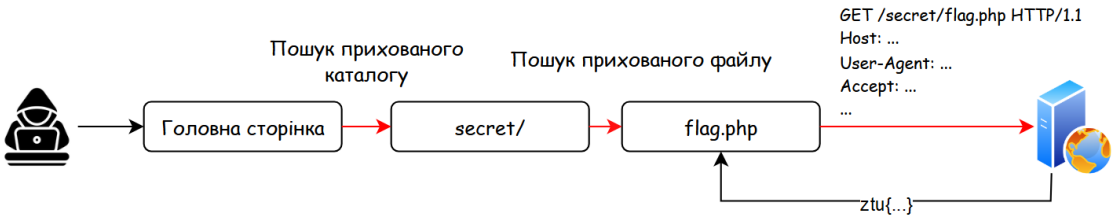


Рисунок 2.2.1 – Графічне зображення дій, необхідних для отримання прапора в першому завданні

Перший веб-сайт (один прапор для захоплення учасниками) – ознайомче завдання, метою якого буде пошук прихованих файлів та каталогів на веб-сайті за допомогою необхідних інструментів. Після знайденого необхідного каталогу “secret/” та файлу всередині цього каталогу “flag.php”, учаснику відобразиться сторінка, на якій буде підказка про необхідність модифікації/додавання певних заголовків в HTTP-запитах від клієнта до веб-сервера. З кожною вірною зміною заголовка учасник отримуватиме нові підказки. Лише у випадку відправлення HTTP-запиту на веб-сервер з усіма необхідними заголовками учасник отримає прапор за успішне виконання завдання.

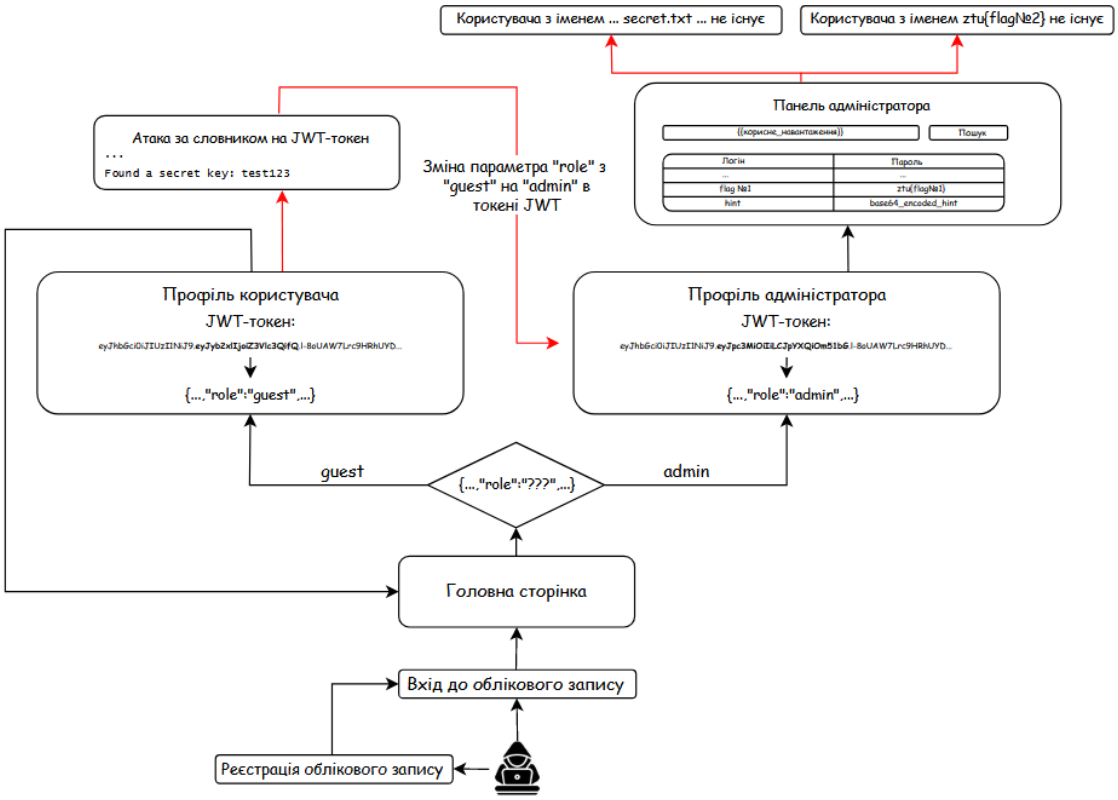


Рисунок 2.2.2 – Графічне зображення логіки веб-сайту та дій, необхідних для захоплення прапорів у другому завданні (зображено червоними стрілками)

Другий веб-сайт (два прапора для захоплення учасниками) – вразливість у використанні слабкого секретного ключа JSON Web (JWT) та Server-Side Template Injection.

Учаснику відображається початковий екран з можливістю входу в обліковий запис, або його реєстрації. Після успішної реєстрації та аутентифікації, користувач автоматично перенаправляється на сторінку свого профілю, де буде залишена підказка, яка вкаже учаснику на те, що обліковий запис повинен мати роль “admin” для розширення функціоналу та отримання першого прапора. Використавши дану підказку, учасник повинен перейти до панелі розробника у браузері, або скористатись іншими інструментами, які покажуть те, що на веб-сайті використовується JWT-токен. Виконавши дешифрування токена за допомогою Base64, учасник дізнається, що наразі для його облікового запису встановлена роль “guest”.

Передбачається, що веб-сайт буде захищений від атак на токен, метою яких є зміна значення алгоритму шифрування на “none”, тому єдиним доступним варіантом для зміни ролі облікового запису – атака за словником для підбору секретного ключа, який використовується для захисту від підробки токена JWT [12]. Після успішного проведення атаки за словником за допомогою спеціального інструмента, учасник отримує секретний ключ, який надасть змогу виконати зміну параметрів в тілі токена (зміна ролі “guest” на роль “admin”).

Після успішної зміни ролі облікового запису, в “хедері” веб-сайту з’явиться надпис “Панель адміністратора”, перейшовши за яким, учаснику виводиться таблиця з усіма логінами та паролями користувачів, серед яких буде користувач з логіном “flag №1” та паролем у вигляді прапора, що свідчитиме про те, що перший прапор успішно захоплений.

Ще один функціонал на веб-сайті – пошук користувача за його логіном за допомогою спеціального поля на веб-сайті, яке буде вразливе до атаки SSTI. Також у таблиці знаходитиметься користувач з логіном “hint”, в паролі якого буде підказка до захоплення другого прапора у вигляді закодованого повідомлення за допомогою Base64 – “Неіснуючий користувач, Jinja2”. Використавши підказку “Неіснуючий користувач”, учасник спробує ввести в поле пошуку користувачів будь-яке

ім’я, якого немає в таблиці, після чого на екрані з’явиться повідомлення: “Користувач з іменем введене_ім’я не існує”. Підказка “Jinja2” свідчитиме про те, що на даному веб-сайті використовується шаблонізатор Jinja2, який вразливий до атаки Server-Side Template Injection, якщо введені користувачем дані не фільтруються. Перевіривши дану теорію вводом {{7*7}} (або будь-яким іншим значенням) та отримавши повідомлення “Користувач з іменем 49 не існує”, учасник створює корисне навантаження (payload) для виконання команд на сервері, після чого, такими командами як “find”, “ls” та “cat”, користувач знайде та відобразить на веб-сайті другий прапор. Після знаходження двох прапорів завдання вважається успішно виконаним.

Загалом, завдання має два прапори для захоплення, що робить його складнішим, ніж завдання з одним прапором, а також потребує знання двох різних типів вразливостей: JSON Web Token (JWT) та Server-Side Template Injection (SSTI) та інструментарію, який необхідний для вирішення даного завдання. Також, учаснику необхідно дослідити веб-сайт, щоб зрозуміти, як він використовує JWT та Jinja2, що може включати перегляд вихідного коду, або використання інструментів для веб-розвідки.

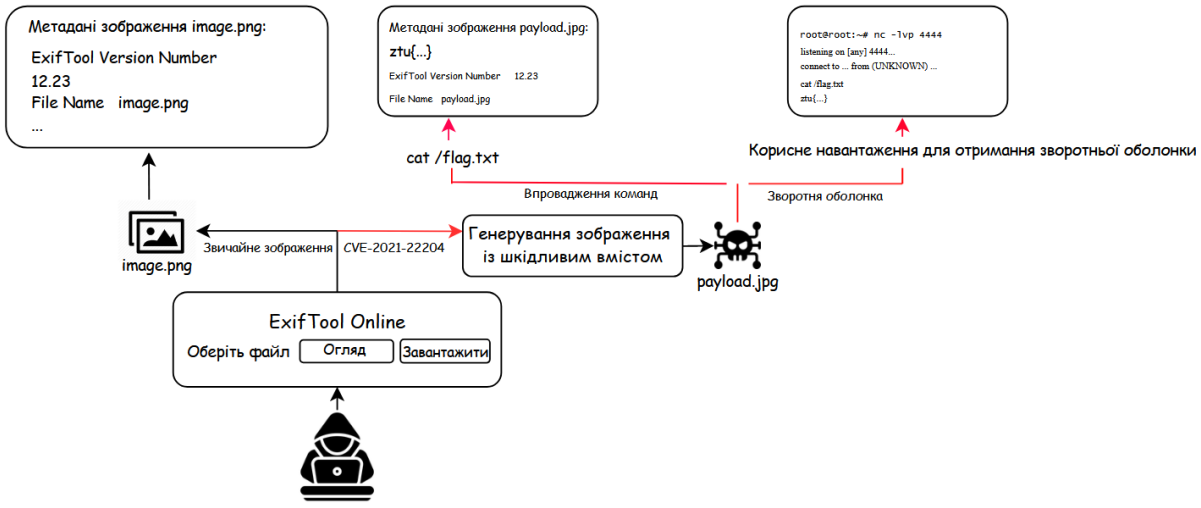


Рисунок 2.2.3 – Графічне зображення логіки веб-сайту та дій, необхідних для захоплення прапора у третьому завданні (зображено червоними стрілками)

Третій веб-сайт (один прапор для захоплення учасниками) - вразливість CVE-2021-22204 (Неналежна нейтралізація даних користувача у форматі файлу DjVu в ExifTool версії 7.44 і вище дозволяє виконувати довільний код під час аналізу шкідливого зображення).

Djvu — це комп'ютерний формат файлів, призначений головним чином для зберігання відсканованих документів, особливо тих, що містять поєднання тексту, лінійних малюнків, індексованих кольорових зображень і фотографій. GitLab описав проблему як автентифіковану вразливість, яка є результатом передачі наданих користувачами зображень у вбудовану версію служби ExifTool [5]. Віддалений зловмисник міг виконувати довільні команди від імені користувача git через неправильну обробку ExifTool файлів DjVu, проблему, якій пізніше було присвоєно CVE-2021-22204.

Перед учасниками відображається веб-сайт "Exiftool Online", який дозволяє завантажувати зображення у форматі jpg, jpeg, png та отримувати відповідь від сервера – детальну інформацію про метадані зображення.

На даному етапі в учасників можуть виникнути думки про два можливих напрямки для атак – завантаження зворотньої оболонки за допомогою шкідливого коду написаного на мові PHP, або виконання атаки Command Injection за допомогою впровадження команд (UNIX) в назву зображення, що завантажується на сервер. Проте дані методи атак не спрацюють через належну фільтрацію типів файлів, що завантажуються (лише png, jpg та jpeg), а також назви файлів (забороняються будь-які символи, що можуть використовуватись для впровадження команд).

В даному завданні виявити вразливість на веб-сайті допоможе уважність та вміння аналізувати учасниками отриману інформацію від сервера. При завантаженні звичайного зображення, користувач у виводі метаданих отримує версію інструмента Exiftool, що використовується на веб-сервері: ExifTool Version Number 12.23. Виконавши пошук вразливостей за даною версією Exiftool (12.23) за допомогою будь-якої пошукової системи, учасник одразу знайде вразливість CVE-2021-22204 - неналежна нейтралізація даних користувача у форматі файлу DjVu в

ExifTool версії 7.44 і вище дозволяє виконувати довільний код під час аналізу шкідливого зображення.

Здійснивши детальніший пошук, учасник знайде готовий експлойт (exploit) або використає інструмент Metasploit, який автоматично згенерує шкідливе зображення з корисним навантаженням, відповідно до потреб (віддалене виконання коду, або отримання зворотньої оболонки від сервера до машини атакуючого). Суть полягає в тому, що скрипт автоматично впроваджує шкідливий код в метадані зображення та при виконанні команди *exiftool назва_шкідливого_зображення*, на веб-сайті одразу виконається код із метаданих зображення.

Після успішної генерації такого зображення та його завантаження на веб-сервер через веб-сайт, користувач або за допомогою команд знайде на сервері файл з прапором та виведе його на екран, або отримає зворотню оболонку, де також командами знайде та відобразить прапор.

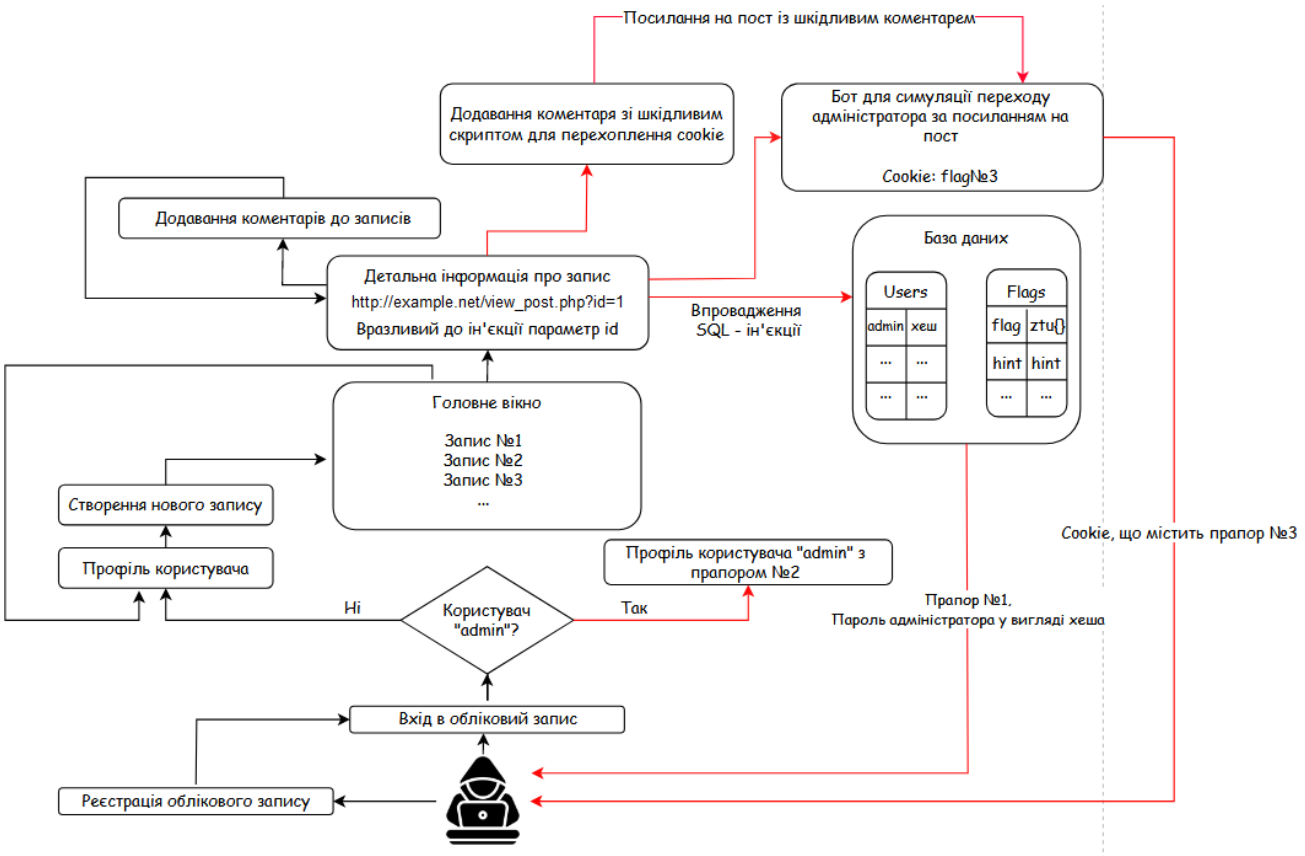


Рисунок 2.2.4 – Графічне зображення логіки веб-сайту та дій, необхідних для захоплення прапорів у четвертому завданні (зображено червоними стрілками)

Четвертий веб-сайт (три прапора для захоплення учасниками) – вразливості SQL-Injection, використання слабких паролів, що зберігаються у вигляді хешів, Cross-Site Scripting (XSS).

Учасникам доступний веб-сайт у вигляді форуму, де для початку необхідно створити обліковий запис та згодом увійти до нього. Веб-сайт використовуватиме базу даних для зберігання даних. Для ускладнення завдання, процес входу до облікового запису буде захищено від впровадження ін'єкцій SQL для входу в обліковий запис адміністратора. Учасник матиме змогу відвідати сторінку свого профіля, головну сторінку, де будуть відображенні усі пости від усіх користувачів, а також переглядати детально кожний пост разом з коментарями, залишеними під ним, на окремій сторінці.

На веб-сайті попередньо буде створений пост на тему баз даних, які краще всього використовувати при розробці. При його перегляді в коментарях знаходитиметься коментар від користувача "admin", в якому буде зазначено, що даний веб-сайт успішно працює з базою даних MySQL. Це дасть учаснику розуміння того, що перший прапор можна отримати впровадивши ін'єкцію, тому перша задача для отримання прапору – знаходження вразливих параметрів для впровадження SQL-ін'єкції.

Переглядаючи попередньо створений пост, учасник побачить, що в URL-посиланні на запис використовується параметр id в який передається унікальний ідентифікатор запису, який і є вразливим до впровадження ін'єкції.

Після знаходження вразливого місця на веб-сайті, учасник повинен використати інструмент, який автоматично впроваджуватиме ін'єкції в запити, допоки не знайде успішний шкідливий запит для отримання даних із бази даних.

Після виконання вищеописаних дій, учасник знайде попередньо створену таблицю "flags", в якій буде залишено перший прапор. В таблиці, в якій зберігаються дані про користувачів, учасник знайде обліковий запис адміністратора з його логіном та паролем, що зберігається у вигляді хешу. Також таблиця "flags" міститиме в собі підказку, яка скерує користувача на думку, що необхідно увійти до облікового запису користувача "admin".

Наступний етап – злам хешу пароля користувача “admin” за допомогою відповідних інструментів. Після успішного зламу хеша та входу на веб-сайті під обліковими даними адміністратора, учасник на сторінці профілю отримає другий прапор.

Для отримання третього прапору учасник отримує підказку на сторінці користувача “admin”, в якій зазначається, що на веб-сайті функціонує бот для модерації записів та коментарів, в який необхідно відправити посилання на пост, на який учасник хоче відправити скаргу, після чого спеціальний користувач нібито перейде до цього поста за посиланням. Також в підказці буде зазначено, що даний спеціальний користувач містить в собі спеціальний “cookie”, в якому міститься прапор. Це має наштовхнути учасника на думку, що необхідно впровадити шкідливий скрипт для експлуатації вразливості XSS для отримання “cookie” даного користувача. Логіка впровадження шкідливого скрипту має бути такою:

– Учасник створює PHP-скрипт, при звертанні до якого будуть передаватися “cookie” кожного, хто звернеться до цього файлу.

– Учасник запускає PHP-сервер, який також згенерує URL посилання на робочу станцію атакуючого, для того, щоб реалізувати віддалене звертання до створеного PHP-файлу.

– На веб-сайті створити пост на будь-яку тему, після чого в коментарі додати шкідливий скрипт, який перенаправлятиме кожного, хто відкриє даний пост, на посилання (на створений PHP-скрипт), яке учасник залишить в своєму скрипті, а також збиратиме cookie цих користувачів.

– Скопіювати посилання на пост, в якому міститься коментар із шкідливим скриптом.

– Надіслати посилання на даний пост в бота на веб-сайті, після чого отримати відповідь на свій PHP-сервер із cookie спеціального користувача, в якому міститиметься третій прапор.

Після проведення усіх вищеописаний дій та отримання трьох прапорів, завдання вважатиметься успішно виконаним.

2.2.2. Визначення рівня складності завдань.

З огляду на дії, які учасник повинен виконати для захоплення прапора в першому випробуванні, завдання відноситься до 1-го рівня складності:

– Завдання не потребує знання жодних вразливостей або складних інструментів. Його можна вирішити за допомогою базових навичок пошуку прихованих файлів та каталогів на веб-сайтів та HTTP-запитів.

– Учаснику не потрібно проводити глибоке дослідження веб-сайту або його коду.

– Завдання не потребує знання програмування або інших спеціалізованих навичок. Достатньо знати інструменти для пошуку прихованих файлів та каталогів, перехоплення та модифікації HTTP-запитів при відправленні на веб-сервер.

Загалом, це завдання є чудовим стартом для учасників, які хочуть ознайомитися з основами веб-безпеки.

Друге завдання з вразливостями секретного ключа JWT-токена та Server-Side Template Injection можна віднести до 3-го рівня складності:

– Завдання потребує знання двох різних типів вразливостей, для експлуатації яких потрібні спеціальні інструменти та навички.

– Учасникам необхідно дослідити веб-сайт, щоб зрозуміти, як він використовує JWT та Jinja2. Це може включати перегляд вихідного коду або використання інструментів для веб-розвідки.

– Завдання вимагає розуміння механізмів JWT та SSTI, а також того, як їх можна експлуатувати.

– Для успішного вирішення завдання учасникам, ймовірно, знадобляться знання програмування для створення корисного навантаження (payload) для SSTI-атаки.

Хоча задача не потребує глибоких знань про певні CVE або невідомі вразливості, вона все ж таки вимагає значного рівня розуміння та навичок. Загалом, це завдання є чудовим викликом для учасників, які прагнуть поглибити свої знання в галузі веб-безпеки.

Виходячи з опису третього завдання в п. 2.2.1, воно потребує наступних знань та навичок:

– Завдання потребує розуміння вразливості CVE-2021-22204, яка стосується неналежної нейтралізації даних користувача у форматі файлу DjVu в інструменті ExifTool.

– Для вирішення завдання учасникам необхідно провести онлайн-дослідження, щоб знайти інформацію про вразливість та способи її експлуатації.

– Хоча знання програмування не є обов'язковими, вони можуть бути дуже корисними при генеруванні шкідливого зображення та використанні експлойту.

З огляду на вищеописані пункти, третє завдання можна віднести до 3-го рівня складності.

Четверте завдання, в якому використовуються вразливості SQL-Injection, використання слабких паролів та XSS, можна віднести до 4-го рівня складності:

– Завдання потребує глибокого розуміння трьох різних вразливостей: SQL-Injection, слабкі паролі та XSS.

– Учасникам необхідно експлуатувати всі три вразливості для отримання прапорів. Це вимагає значних знань та навичок в обході веб-безпеки.

– Учасникам необхідно знати та вміти використовувати різні інструменти для атаки, такі як інструменти для SQL-Injection та зламу хешів.

– Знання програмування необхідні для створення шкідливого PHP-скрипту для експлуатації вразливості XSS.

– Учасникам необхідно адаптувати свої знання та навички до конкретного контексту завдання.

– Завдання ґрунтується на експлуатації вразливостей RCE (SQL-Injection) та XSS, які є більш складною, ніж вразливості рівня 1, 2 або 3.

Отже, на етапі проектування було визначено, що завдання будуть мати наступні рівні складності:

– Завдання №1 (пошук прихованих файлів/каталогів та зміна заголовків в HTTP-запитах) – 1-й рівень складності.

– Завдання №2 (JWT та SSTI) – 3-й рівень складності.

– Завдання №3 (CVE-2021-22204) – 3-й рівень складності.

– Завдання №4 (SQLi, злам хешу, XSS) – 4-й рівень складності.

2.3. Вибір інструментів та технологій для створення веб-сайтів та реалізації завдань.

2.3.1. Середовище виконання.

Передбачається, що кожен веб-сайт та завдання розроблятимуться на окремих віртуальних машинах з подальшим їх встановленням та підключенням до локальної мережі учасниками для зручності та швидкості розгортання.

Як платформу віртуалізації було обрано VirtualBox. VirtualBox – це безкоштовна та відкрита платформа віртуалізації, яка дозволяє створювати та запускати віртуальні машини (ВМ) на одному фізичному комп'ютері. Її простота використання, гнучкість та висока продуктивність роблять VirtualBox популярним вибором як для домашнього, так і для професійного використання.

Переваги використання VirtualBox для змагань формату CTF:

– Кожна ВМ працює в власному ізольованому середовищі, що гарантує, що проблеми на одній ВМ не вплинуть на інші. Це важливо для CTF, де учасники можуть запускати шкідливий код, не ризикуючи пошкодити свої основні системи.

– ВМ можна легко копіювати та переміщати між комп'ютерами, що робить їх зручними для спільного використання та розгортання.

– VirtualBox підтримує широкий спектр операційних систем, такі як Linux, Windows, macOS та Solaris. Це дозволяє учасникам CTF використовувати будь-яку ОС, необхідну для виконання завдань.

– VirtualBox має простий та інтуїтивно зрозумілий графічний інтерфейс користувача, що робить його доступним для користувачів з будь-яким рівнем досвіду.

– VirtualBox може забезпечити високу продуктивність ВМ, що важливо для ресурсомістких завдань CTF.

Для кожного веб-сайту та завдання буде створена окрема віртуальна машина. На кожну ВМ буде встановлена необхідна операційна система та програмне забезпечення. Для розгортання віртуальних машин на робочій станції, учасники повинні спочатку завантажити ВМ із необхідним завданням, після чого додати її до локальної мережі (надати IP-адресу за допомогою створеної мережі NAT всередині Virtu-

alBox, або шляхом додавання до побудованої мережі в емуляторі GNS). Після проведених налаштувань, учасники отримають доступ до веб-сайтів за допомогою IP-адрес або імен хостів.

Платформою для веб-сервера та розробки веб-сайтів оберемо операційну систему Linux, а саме її дистрибутив Ubuntu версії 22.04. Це зробить процес написання коду для веб-сайтів набагато зручнішим завдяки графічному інтерфейсу, який надає даний дистрибутив. Також Ubuntu відомий своєю стабільністю та надійністю, що робить його чудовим вибором для критично важливих веб-застосунків, які потребують безперебійної роботи. Також виділимо додаткові переваги Ubuntu:

– Ubuntu має широкий спектр інструментів для веб-розробки, включаючи мови програмування, фреймворки та системи керування контентом (CMS). Це робить його зручним середовищем для розробки веб-сайтів будь-якої складності.

– Ubuntu має велику та активну спільноту користувачів.

– Ubuntu поставляється з безліччю функцій безпеки, які допоможуть захистити веб-сервер від небажаних атак.

– Ubuntu - це безкоштовний дистрибутив Linux, що робить його доступним для будь-якого користувача.

Для розгортання веб-сайту оберемо веб-сервер Apache HTTP Server. Apache – це популярний веб-сервер з відкритим кодом, який широко використовується в Інтернеті. Такі характеристики як гнучкість, масштабованість та надійність роблять даний веб-сервер гарним вибором для розгортання веб-сайтів CTF. Для розробників корисним є те, що існує безліч ресурсів з документації Apache, що допоможе при створенні веб-сайтів. Конфігураційні файли Apache відносно прості для розуміння та налаштування, що дозволяє швидко розгорнути вразливі веб-сайти та змінювати їх конфігурацію в залежності від завдань змагань. Також, Apache працює на більшості операційних систем, включаючи Unix, Linux, Windows і macOS. Це дозволяє розгорнути сервер у будь-якому середовищі, яке зручне для організаторів змагань.

З точки зору продуктивності, вибір стоїть між веб-серверами Apache та Nginx. Nginx відомий своєю високою продуктивністю та ефективністю при роботі з великими обсягами одночасних з'єднань, проте передбачається, що до розроблених завдань в даній роботі не буде відбуватись одночасне з'єднання багатьох учасників, адже завдання розраховані на одну або декілька осіб, які будуть під'єднуватися до сервера за допомогою локальної мережі, тому в цьому контексті також краще обрати веб-сервер Apache.

У контексті безпеки, Apache за замовчуванням використовує користувача та групу "www-data", який зазвичай має мінімальні привілеї та права доступу лише до файлів веб-сайту, що забезпечує ізоляцію процесів Apache від інших процесів і користувачів системи. Це знижує ризик того, що зловмисник, який отримає доступ до веб-сервера, зможе вплинути на інші частини системи, окрім тих, які передбачені розробником завдань у форматі CTF.

Серед мінусів сервера Apache виділяють вразливості, що можуть виникнути внаслідок зміни конфігурації, або використання застарілих версій. Проте, при розробці завдань у форматі CTF або загалом за напрямком Penetration Testing, це може зіграти на руку організаторам для демонстрації та експлуатації вразливостей, що можуть виникати внаслідок вищеописаних дій.

Як інструмент для розробки оберемо "VSCodium". VSCodium - це безкоштовний редактор коду з відкритим кодом, який є чудовим вибором для розробників Він пропонує широкий спектр функцій, які роблять його потужним та зручним у використанні, а також має кілька переваг перед своїми конкурентами.

Серед ключових особливостей Codium можна виділити наступні:

– Codium підтримує широкий спектр мов програмування, включаючи Python, Java, JavaScript, C++, C#, мову розмітки HTML та мову стилів CSS, PHP та багато інших.

– Codium можна налаштувати за допомогою тем, шрифтів та інших параметрів, щоб створити робоче середовище, яке відповідає потребам розробника.

– Codium є безкоштовним та з відкритим кодом, що робить його доступним для будь-якого користувача.

– Codium не споживає багато системних ресурсів.

– Codium доступний для різних операційних систем, включаючи Ubuntu, Windows та macOS.

– Codium сумісний з розширеннями та темами VS Code, що робить його зручним для переходу з VS Code.

Загалом, редактор коду Codium є гарним вибором для написання коду для веб-сайтів у контексті створення завдань у форматі CTF, використовуючи дистрибутив Ubuntu.

2.3.2. Вибір мов програмування та веб-фреймворків для розробки веб-сайтів.

Для першого веб-сайту CTF, де учасникам потрібно знайти приховані файли та каталоги, а потім модифікувати HTTP-заголовки для отримання прапора, PHP є найкращим вибором мови програмування з наступних причин:

– PHP - це широко використовувана мова програмування, яка легко вивчається та доступна. Більшість веб-хостингів підтримують PHP за замовчуванням, що робить її зручною для розгортання CTF-завдань.

– PHP тісно інтегрована з веб-серверами Apache та Nginx, що робить її ідеальною для створення динамічних веб-сторінок та обробки HTTP-запитів.

– PHP надає прості у використанні функції для доступу, модифікації та додавання HTTP-заголовків у запити та відповіді, що робить його зручним для створення такого типу завдань, де учасникам потрібно модифікувати заголовки HTTP-запитів для отримання прапора.

Для розробки другого веб-сайту з вразливостями JWT-токена та SSTI використаємо наступний стек технологій: Python, Flask, Jinja2.

По-перше, Python є високорівневою мовою програмування, яка забезпечує зручність та швидкість розробки завдяки своєму чистому синтаксису та багатому набору бібліотек. Це робить Python популярним вибором для створення веб-додатків та прототипів.

Flask, мікрофреймворк для Python, є ідеальним інструментом для створення веб-додатків з мінімальною кількістю налаштувань. Flask надає простий та зрозумілий інтерфейс для розробки веб-сервісів, що дозволяє зосередитися на логіці

завдання без необхідності занурюватися в складнощі конфігурації. Крім того, Flask підтримує розширення, що дозволяє додавати функціонал за потреби, зберігаючи основну структуру додатку простою та зручною для підтримки.

Jinja2, потужний шаблонізатор для Python, є частиною стандартного стеку Flask. Jinja2 забезпечує гнучкість та зручність створення динамічних HTML-шаблонів. Однак, якщо дані, введені користувачем, не фільтруються належним чином, Jinja2 може бути вразливим до атак типу SSTI. Використання Jinja2 у даному контексті дозволяє легко ілюструвати вразливості, пов'язані з SSTI, оскільки учасники можуть вводити дані в шаблони, що виконуються на сервері, і таким чином маніпулювати вихідним кодом.

Для демонстрації вразливості JWT, Flask дозволяє легко інтегрувати JWT-токени для аутентифікації та авторизації користувачів. Це забезпечує учасників завдання можливістю переглянути і модифікувати свої токени, вивчивши механізми їх створення та верифікації.

Отже, вибір Python, Flask та Jinja2 для даного завдання є оптимальним через зручність і швидкість розробки, гнучкість у створенні веб-сервісів, та можливість легкої демонстрації і експлуатації вразливостей типу JWT та Server-Side Template Injection.

Для розробки третього та четвертого завдань використаємо мову програмування PHP. Пріоритетом для розробки третього завдання є те, що PHP забезпечує потужні можливості для роботи з командним рядком на сервері. Це дозволяє легко викликати зовнішні програми, такі як ExifTool, безпосередньо з PHP-коду, що є необхідним для демонстрації вразливості CVE-2021-22204. У даному випадку, веб-додаток на PHP може приймати зображення від користувачів, викликати ExifTool для аналізу метаданих та виводити результати на веб-сторінку.

Для реалізації четвертого завдання мова програмування PHP ідеально підходить з декількох причин:

– PHP тісно інтегрується з базами даних, такими як MySQL, що робить його ідеальним вибором для динамічних веб-сайтів, які потребують доступу та обробки інформації з бази даних.

– PHP забезпечує зручні засоби для роботи з хешуванням паролів. Вбудовані функції, такі як “password_hash” та “password_verify”, дозволяють безпечно зберігати та перевіряти паролі користувачів. Однак для навчальних цілей можна реалізувати слабкі методи хешування, щоб учасники могли зламати хеші та знайти відповідні паролі.

– PHP дозволяє легко реалізувати захист від SQL-ін'єкцій та інших типів атак. Водночас для демонстрації вразливостей можна створити умисні уразливі місця, такі як недостатньо захищені SQL-запити, що дозволяють учасникам виконувати SQL-ін'єкції.

– Щодо Cross-Site Scripting (XSS), PHP дозволяє легко впроваджувати та обробляти дані, що вводяться користувачами, що робить можливим реалізацію вразливостей XSS. PHP забезпечує гнучкість у роботі з HTML та JavaScript, що дозволяє учасникам вставляти шкідливий скрипт у коментарі або пости для експлуатації вразливості.

– Крім того, PHP-сервери легко налаштовуються і дозволяють швидко розгортати веб-додатки, що важливо для створення тестового середовища для учасників. Мова PHP також підтримує можливість запуску зовнішніх скриптів та обробки HTTP-запитів, що необхідно для реалізації механізмів збору "cookie" через XSS-атаки.

– PHP також має велику і активну спільноту розробників, що означає наявність багатого вибору бібліотек та фреймворків, які можуть значно спростити та прискорити розробку веб-додатків. Це дозволяє використовувати готові рішення для різних задач, таких як аутентифікація, обробка форм, робота з файлами, та багато іншого, що зменшує час розробки та підвищує ефективність процесу.

Таким чином, PHP є оптимальним вибором для розробки завдання з уразливостями SQL-Injection, слабкими хешами паролів та XSS завдяки своїй зручності у роботі з базами даних, можливостям хешування паролів, гнучкості у роботі з HTML та JavaScript, а також широкій підтримці та наявності готових рішень для розробників.

2.3.3. Вибір та проектування бази даних.

В попередніх пунктах було описано, що завданні під номером 4, код буде вразливим до провадженням учасниками SQL-ін’єкції, тому для реалізації оберемо базу даних MySQL, виділивши наступні характеристики:

– MySQL легко встановити та налаштувати на Ubuntu за допомогою apt-get.

– MySQL є однією з найпопулярніших баз даних, тому вона знайома багатьом розробникам та тестувальникам.

– MySQL підтримує різні типи даних та функції, що робить її гнучкою платформою для тестування вразливостей.

– Завдяки великій спільноті користувачів MySQL можна легко знайти допомогу та підтримку.

Так як концепція четвертого завдання – онлайн-форум, необхідно описати базу даних, таблиці та зв’язки між ними. В назві бази даних буде фігурувати прізвище “laichuk”, тому повна назва – “sql_laichuk”. Загалом, функціонал веб-сайту включатиме в собі наступні компоненти:

– Користувачів.

– Записи (пости) на веб-сайті, які зможе залишати будь-хто з користувачів.

– Коментарі до записів, які також зможе залишати будь-хто з користувачів.

З огляду на вищеописану інформацію, в базі даних будуть створені три таблиці з назвами:

– “users”;

– “posts”;

– “comments”.

Також, за описаною логікою завдання у п. 2.2.1, необхідно створити ще одну таблицю, яка міститиме перший прапор, а також підказку для захоплення наступних прапорів. Назва такої таблиці – “flags”.

Таблиця “users” зберігатиме в собі унікальний ідентифікатор з числовим типом даних integer, назву облікового запису (username) та пароль користувача (password) з текстовим типом даних varchar.

Таблиця “posts” міститиме в собі наступні поля:

– унікальний ідентифікатор (id);

– заголовок запису (title);

– вміст запису (content);

– зовнішній ключ – ідентифікатор користувача, що залишив запис;

Таблиця “comments” зберігатиме в собі:

– унікальний ідентифікатор (id);

– коментар (comment);

– зовнішній ключ - ідентифікатор запису, під яким буде залишено коментар (post_id);

– зовнішній ключ – ідентифікатор користувача, який написав коментар (user_id);

Таблиця “flags” мітитиме в собі такі поля як “name” та “data”, де будуть зберігатися прапор та підказка.

Спроектуємо діаграму бази даних “sql_laichuk”, беручи за основу вищеописану інформацію (рис. 2.3.5).

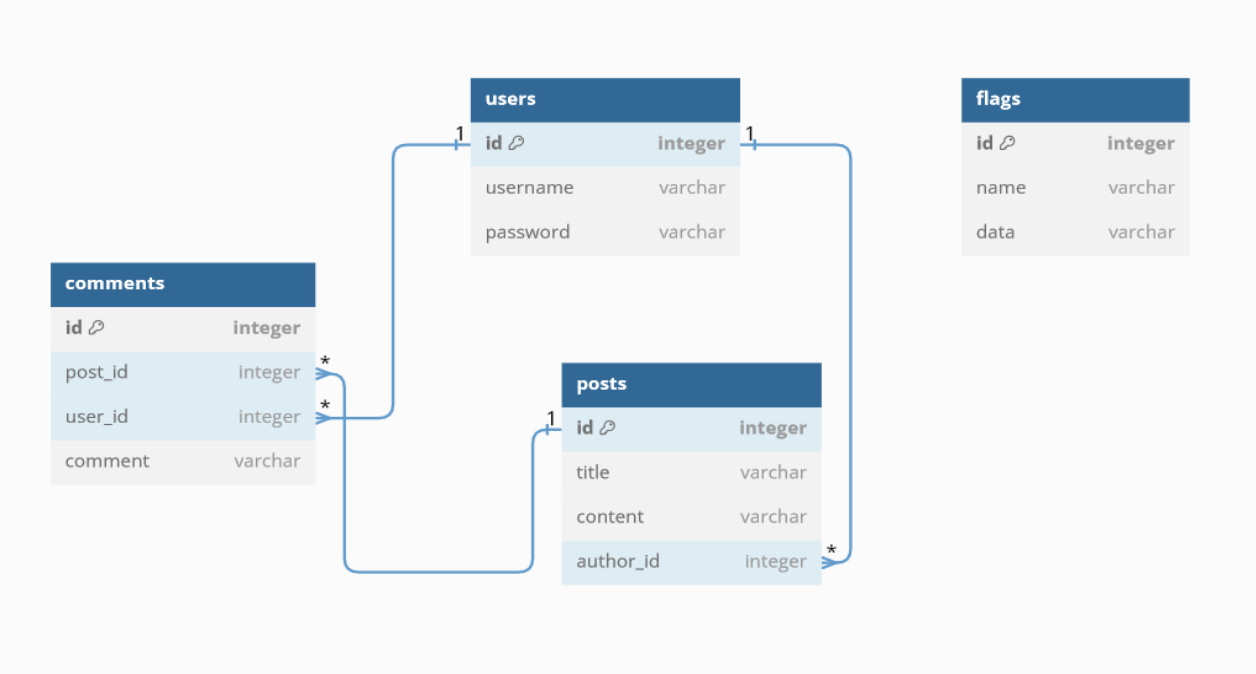


Рисунок 2.3.1 – Діаграма бази даних “sql_laichuk”

На даному етапі база даних успішно обрана та спроектована з детальним описом кожної із таблиць.

2.3.4. Інструменти, що необхідні учасникам для експлуатації вразливостей.

Для успішного вирішення першого завдання, де за мету ставиться пошук прихованих каталогів та файлів з подальшою зміною заголовків в HTTP-запиті до веб-сервера, учаснику необхідно використати деякі з цих інструментів на вибір:

Для пошуку прихованих каталогів та файлів:

– DirBuster - простий у використанні інструмент для брутфорса каталогів, який підтримує словниковий та маскований методи. Він доступний як у безкоштовній, так і в платній версіях - https://www.kali.org/tools/dirbuster/

– dirb - використовує списки слів (словники) для генерації можливих URL-адрес і автоматично перевіряє кожен можливий URL з використанням HTTP-запитів. Результати перебору виводяться в зручному форматі, що дозволяє легко побачити знайдені директорії та файли. Це корисно для тестування безпеки веб-сайтів та виявлення потенційно вразливих точок.

– GoBuster - вільний та відкритий інструмент, який може використовуватися для брутфорса каталогів, файлів та інших ресурсів веб-сайту. Він підтримує різні методи брутфорса, включаючи словниковий, маскований та гібридний. https://github.com/topics/gobuster

Для модифікації заголовків в HTTP-запитах від клієнта до сервера можна використати, наприклад, наступні інструменти:

– Burp Suite - комплексна платформа для тестування проникнення веб-додатків, яка включає в себе потужний інструмент для модифікації заголовків HTTP. Burp Suite дозволяє легко додавати, видаляти та змінювати будь-який заголовок HTTP в запиті. Даний інструмент за замовчуванням встановлено на дистрибутиві Kali Linux.

– curl - це командно-рядковий інструмент для передачі даних, який також можна використовувати для модифікації заголовків HTTP. Щоб змінити заголовок HTTP в curl, використовується прапорець -H. Також даний прапорець можна використовувати для видалення або зміни існуючих заголовків, наприклад, curl -H "User-Agent: MyCustomUserAgent" https://www.example.com.

Успішне виконання другого завдання потребуватиме стороннього інструмента, отримати який можна з відкритих репозиторіїв, для виконання підбору за допомогою словника секретного ключа токена JWT. Таким інструментом, наприклад, може бути Flask-Unsign (https://github.com/Paradoxis/Flask-Unsign) або JWT_Tool (https://github.com/ticarpi/jwt_tool). Обидва інструмента допоможуть:

– Сканувати токен на наявність неправильних конфігурацій або відомих вразливостей.

– Перевірити дійсність секретного/ключового файлу/відкритогоключа/JWKS ключа.

– Виявити слабкі ключі за допомогою високошвидкісної словникової атаки.

– Підробити новий заголовое токену та вмісту корисного навантаження і створення нового підпису за допомогою ключа або іншого методу атаки.

– Підробити мітки часу.

– Генерувати та відновити ключі RSA та ECDSA (з файлів JWKS).

На сторінці репозиторію зазначено, що інструмент JWT_Tool призначений для пентестерів, яким необхідно перевірити надійність токенів, що використовуються, та їх вразливість до відомих атак. Доступні різні варіанти фальсифікації, підписання та перевірки, які допоможуть глибше зануритися в потенційні слабкі місця, наявні в деяких бібліотеках JWT. Він також виявився успішним для учасників CTF, оскільки CTF, схоже, зараз дуже зацікавлені в JWT. Він також може бути корисним для розробників, які використовують JWT в своїх проектах, але хотіли б перевірити стабільність і наявність відомих вразливостей при використанні підроблених токенів.

Для запуску даних інструментів необхідно оновити Python до 3-ї версії.

Виконання атаки SSTI не потребує жодних інструментів.

Для успішного виконання третього завдання учаснику необхідно також встановити сторонній інструмент для автоматичного генерування корисного навантаження у вигляді зображення. Таким інструментом може бути exploit-CVE-2021-22204 (https://github.com/UNICORDev/exploit-CVE-2021-22204) або CVE-2021-22204-exiftool (https://github.com/convisolabs/CVE-2021-22204-exiftool). В описі до

інструмента CVE-2021-22204 зазначено, що експлойт створює корисне навантаження у форматі JPEG, яке можна використати у вразливій версії ExifTool для виконання коду. Можна вказати спеціальну команду або згенерувати зворотню оболонку. Зображення у форматі JPEG генерується автоматично, і, за бажанням, можна надати власне зображення у форматі JPEG для вставки корисного навантаження. Також для експлуатації вразливості CVE-2021-22204 можна використати модуль “exiftool_djvu_ant_perl_injection” за допомогою інструмента Metasploit для того, щоб отримати зворотню оболонку та захопити прапор.

Для захоплення першого прапора в четвертому завданні, користувач може впровадити SQL-ін’єкцію як вручну, так і за допомогою автоматизованого інструмента. Відмінним вибором буде інструмент “sqlmap”.

sqlmap - це безкоштовний та відкритий інструмент для тестування на проникнення, який автоматизує процес виявлення та експлуатації вразливостей SQL-ін'єкцій та захоплення серверів баз даних. Спектр функцій даного інструмента досить широкий:

– sqlmap може автоматично сканувати веб-сайти та веб-додатки на наявність вразливостей SQL-ін'єкцій. Він підтримує різні типи вразливостей SQL-ін'єкцій, включаючи вразливості на основі GET, POST, cookies та заголовків HTTP.

– sqlmap може виконувати довільні SQL-запити на вразливих серверах баз даних. Це дозволяє тестувальникам на проникнення отримувати доступ до даних, які зазвичай недоступні, а також змінювати або видаляти дані.

– sqlmap підтримує широкий спектр баз даних, включаючи MySQL, Oracle, Microsoft SQL Server, PostgreSQL, Microsoft Access, SQLite тощо.

Після отримання доступу до даних, що зберігаються в базі даних, за допомогою sqlmap, учасник також отримає пароль користувача “admin” у вигляді хеша. Пароль попередньо буде досить легким для того, щоб учасник застосував інструмент для виконання атаки за словником для зламу хеша. Такими інструментами можуть бути “hashcat” або “John the Ripper”, що описані в розділі 1 п. 1.2.3.

Для проведення XSS-атаки учасникам не потрібно завантажувати жоден із інструментів, адже передбачається ручне виконання атаки.

РОЗДІЛ 3. ПРАКТИЧНА РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ СТВОРЕНИХ ВЕБ-ДОДАТКІВ

3.1. Підготовка середовища до розробки веб-додатків.

Описані нижче дії будуть виконуватись на кожній із чотирьох віртуальних машин. На початку роботи необхідно оновити інформацію про доступні пакети з усіх підключених репозиторіїв та оновити встановлені пакети до останніх версій, які є в репозиторіях на операційній системі Ubuntu за допомогою команд “sudo apt update” та “sudo apt upgrade -y” (рис. 3.1.1 – 3.1.2).

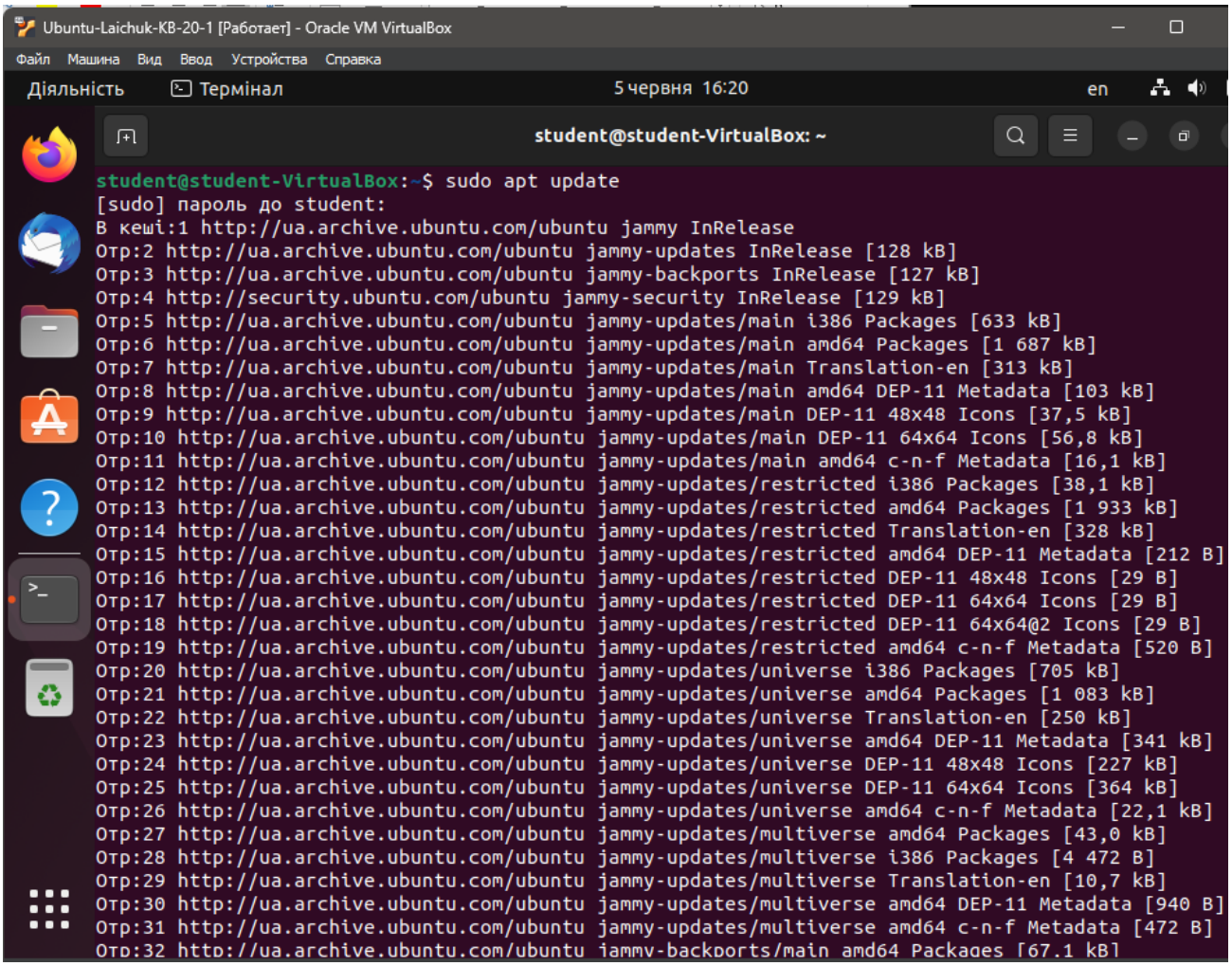


Рисунок 3.1.1 – Успішне оновлення інформації про доступні пакети з усіх підключених репозиторіїв за допомогою команди “sudo apt update”

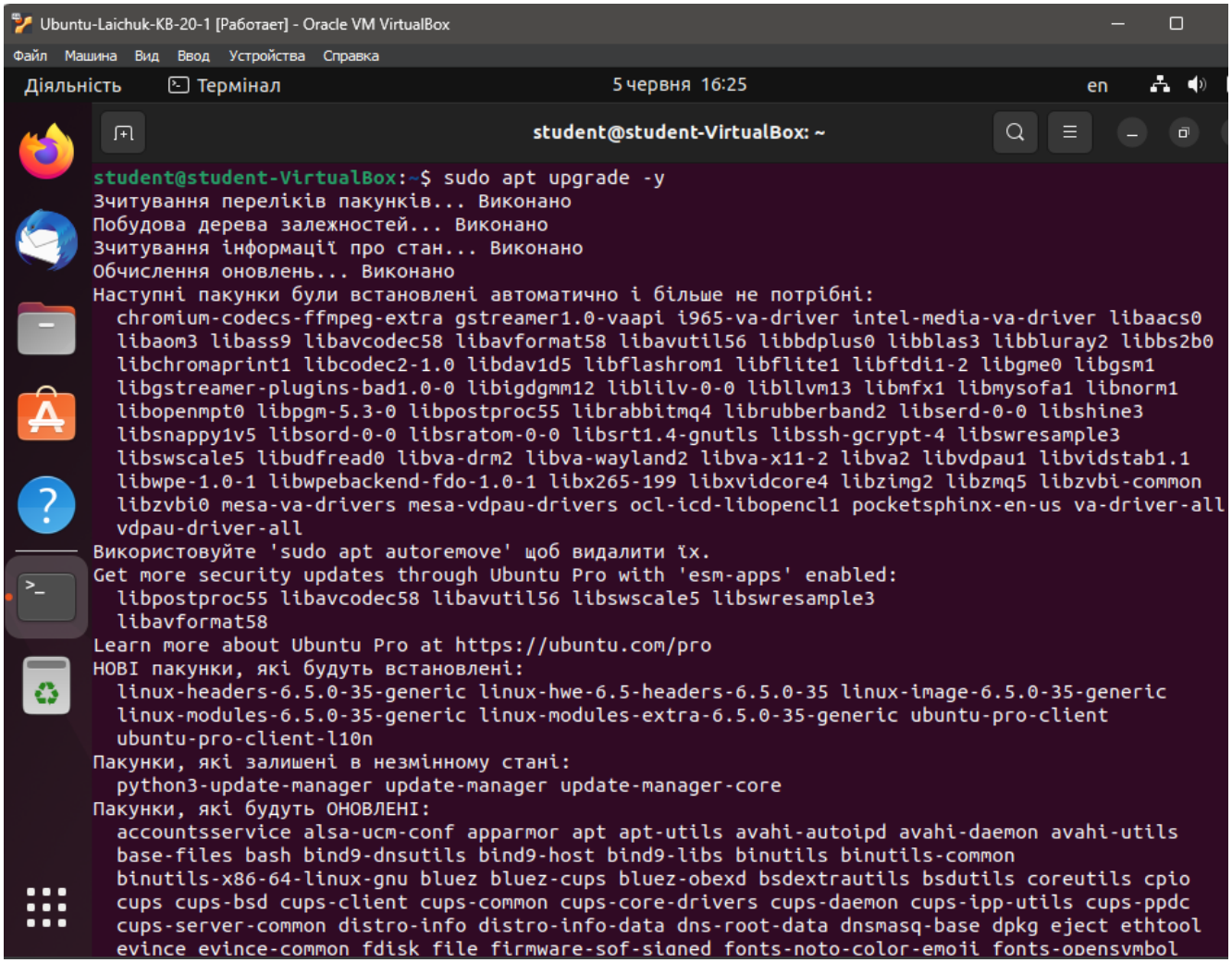


Рисунок 3.1.2 – Процес оновлення встановлених пакетів, що наявні в репозиторіях, до останніх версій, за допомогою команди “sudo apt upgrade -y”

Так як веб-сайти розгортатимуться за допомогою веб-сервера Apache, наступним кроком буде його встановлення (рис. 3.1.3 - 3.1.4). Встановлення виконується за допомогою команди “sudo apt install apache2 -y”. Параметр “-y” в даній команді використовується для автоматичного підтвердження всіх запитів під час встановлення пакета.

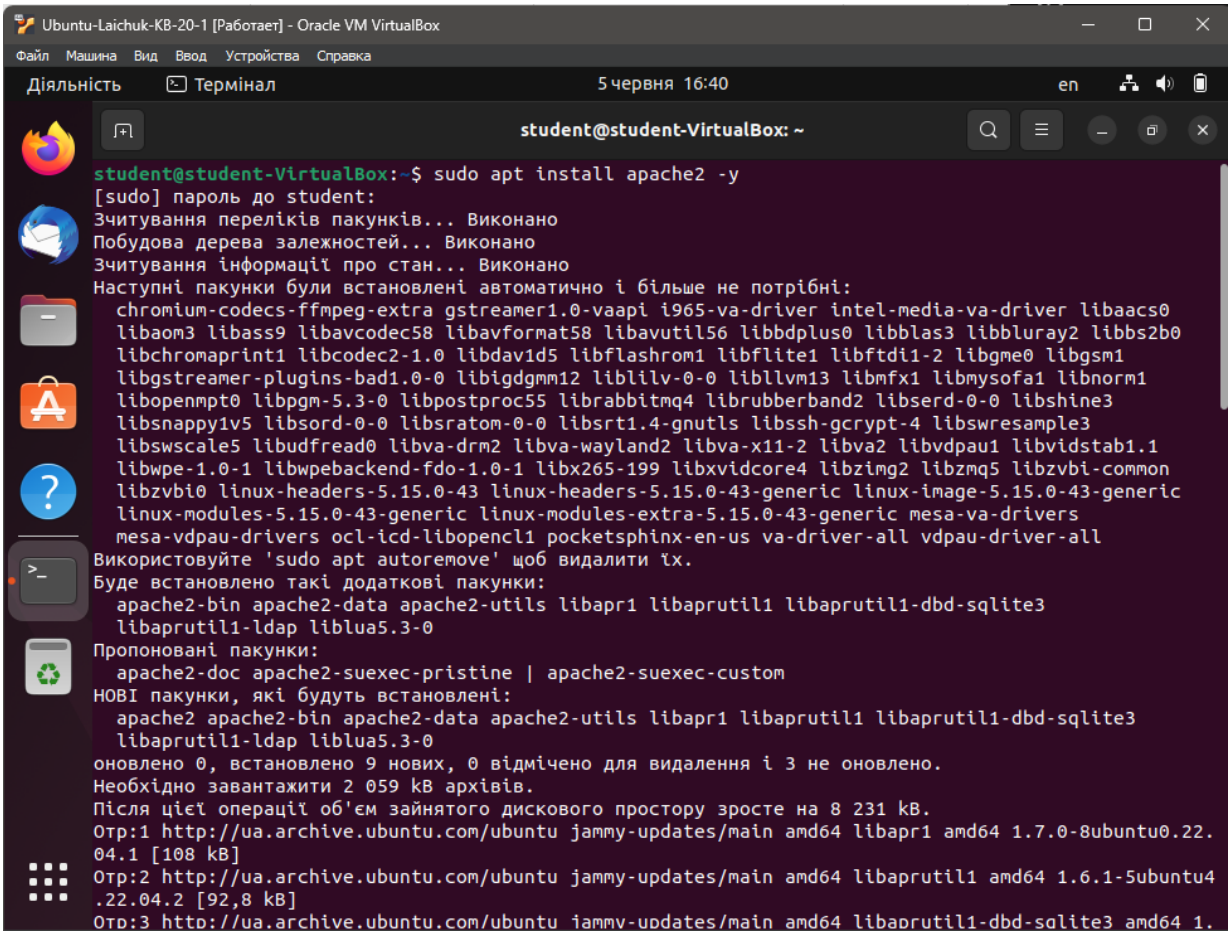


Рисунок 3.1.3 – Процес встановлення HTTP-сервера Apache

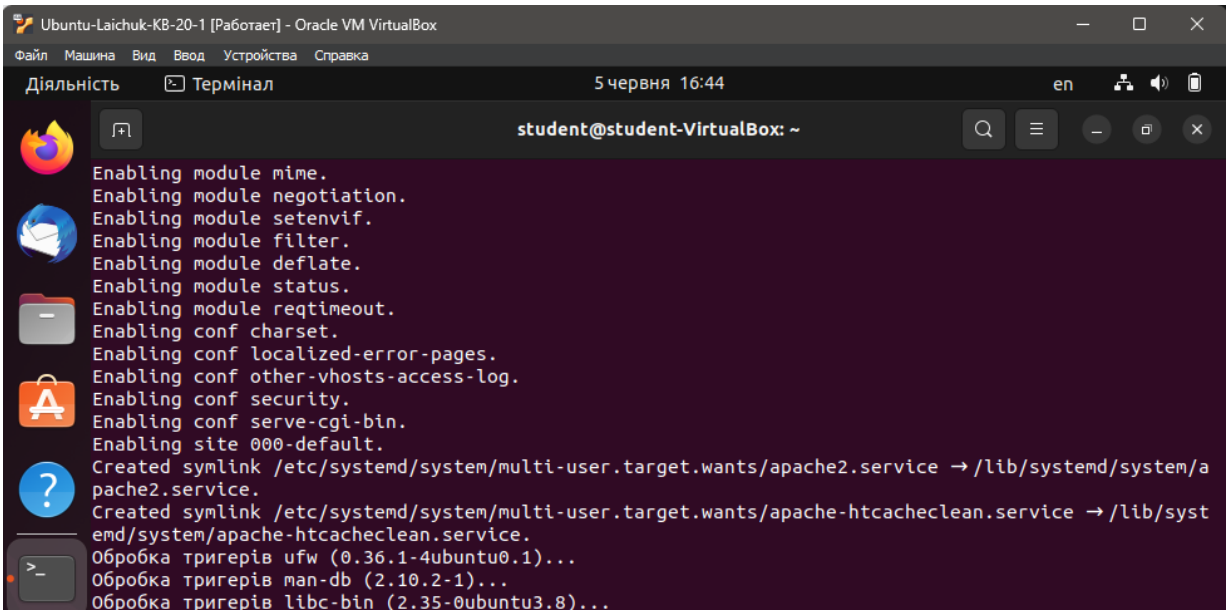


Рисунок 3.1.4 – Успішне встановлення HTTP-сервера Apache

Для сайту, що буде написаний на мові програмування Python та за допомогою веб-фреймворка Flask, необхідно встановити компоненти “python3” та “python3-pip” (рис. 3.1.5 – 3.1.6).

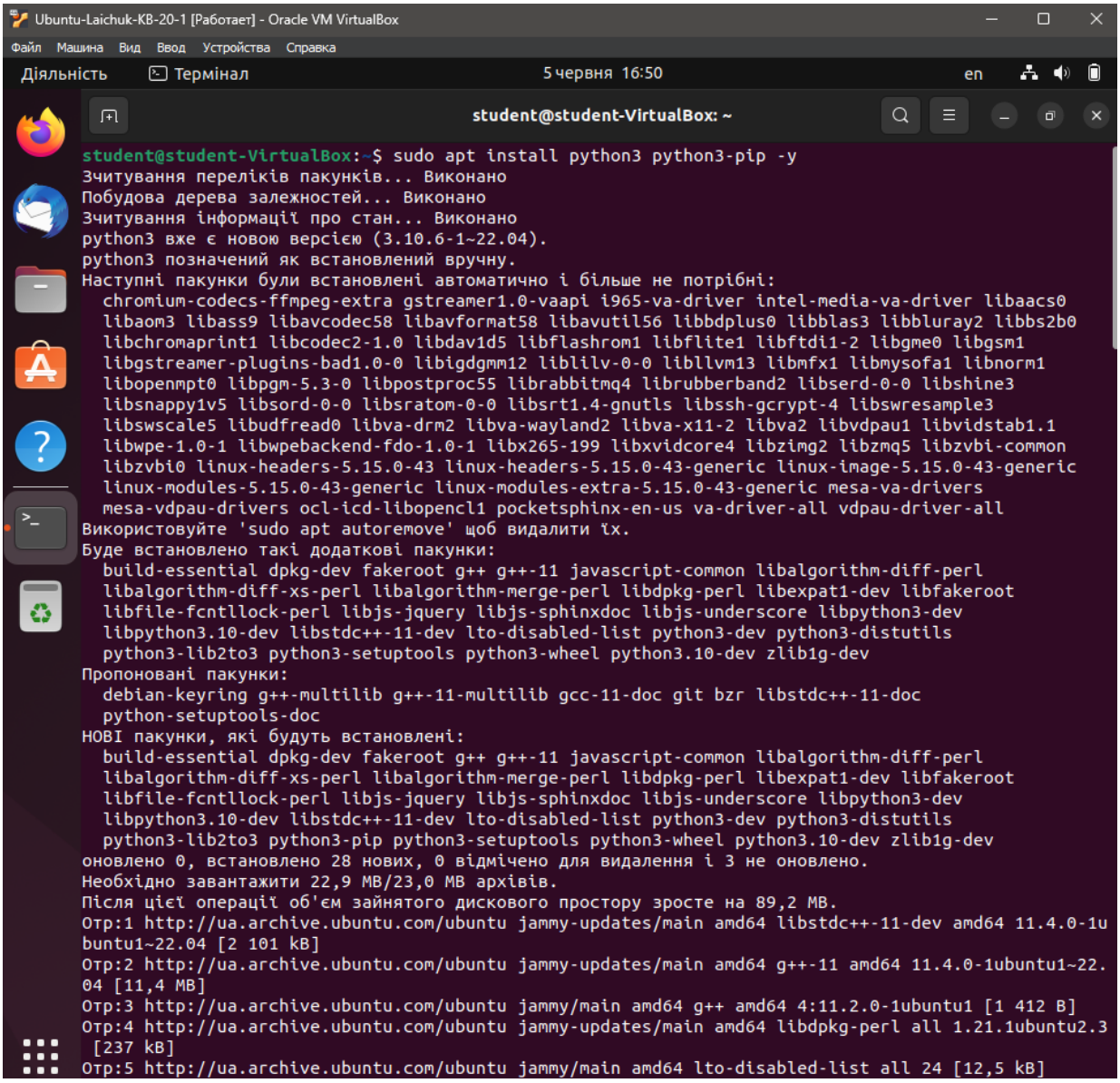


Рисунок 3.1.5 – Процес встановлення компонентів “python3” та “python3-pip”

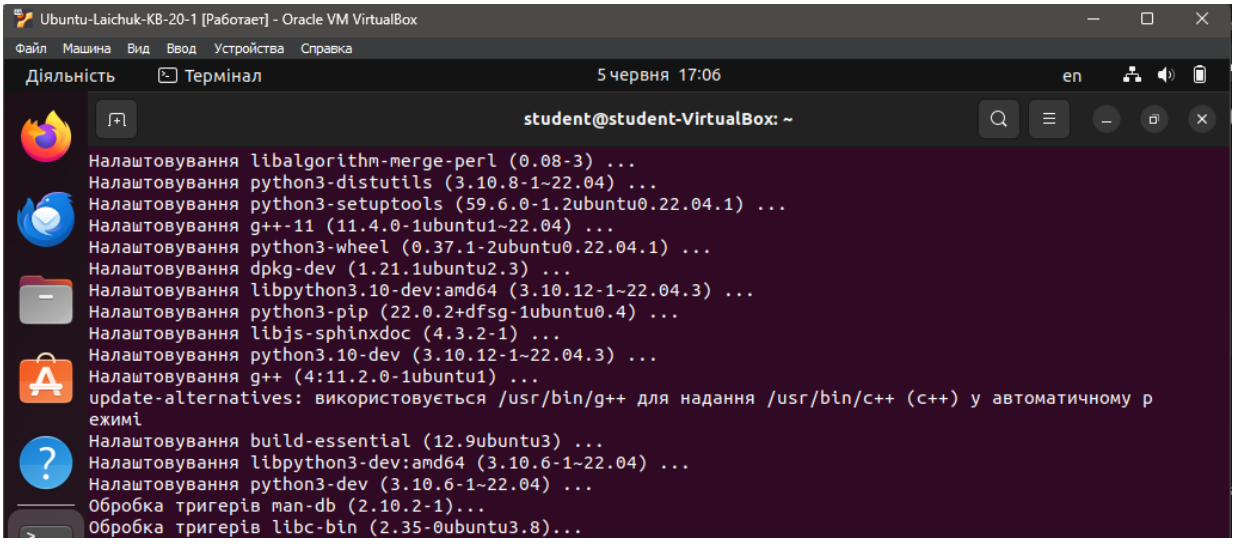


Рисунок 3.1.6 – Успішне встановлення компонентів “python3” та “python3-pip”

Наступним кроком є встановлення модуля “libapache2-mod-wsgi-py3” за допомогою команди “sudo apt install libapache2-mod-wsgi-py3” (рис. 3.1.7). Даний модуль дозволяє запускати веб-додатки, написані на Python 3, з використанням протоколу WSGI (Web Server Gateway Interface).

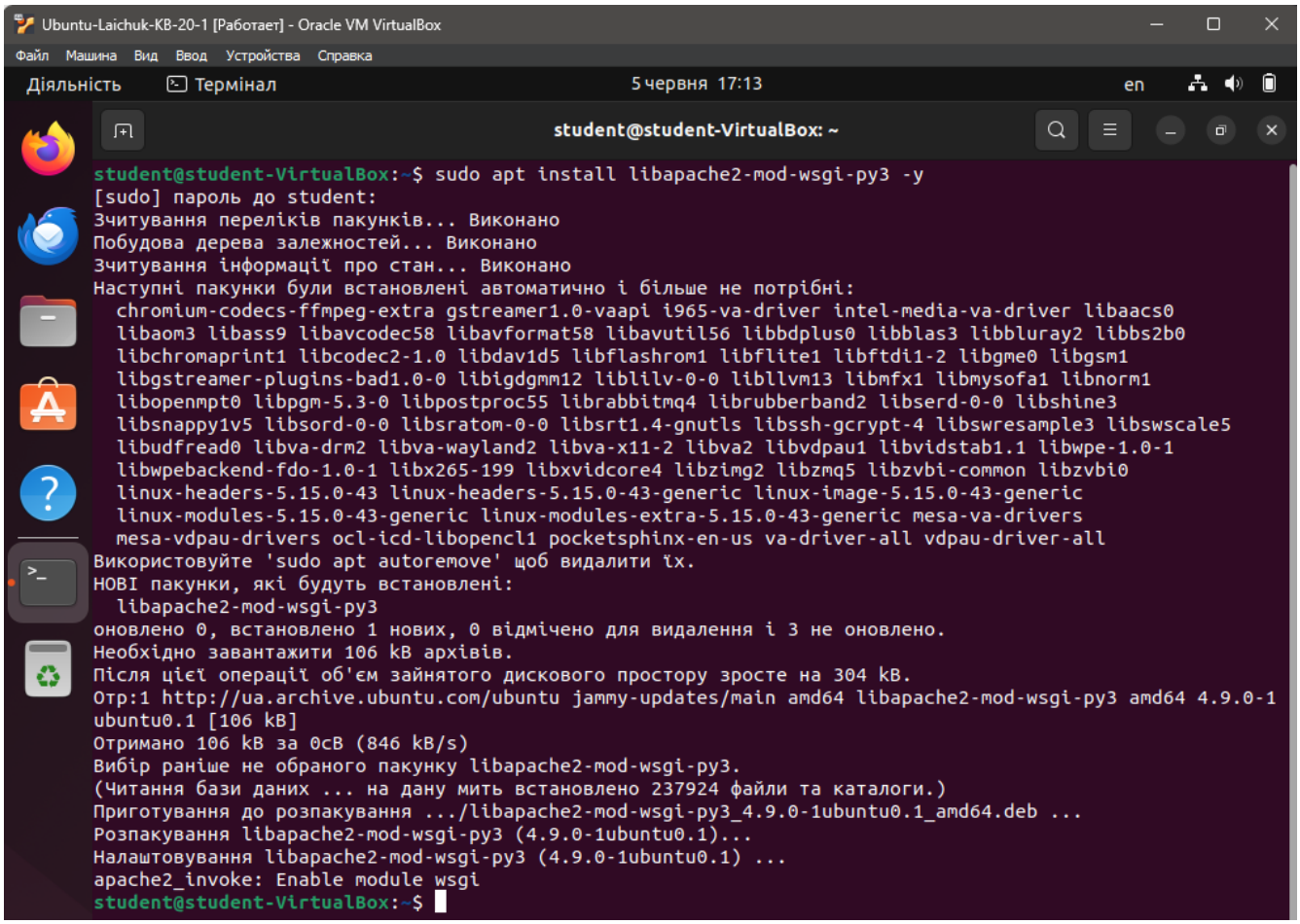


Рисунок 3.1.7 – Успішне встановлення модуля “libapache2-mod-wsgi-py3”

Для встановлення веб-фреймворка Flask для мови програмування Python, спочатку необхідно створити робочий каталог за допомогою команди “mkdir”, після чого перейти до новоствореного каталогу і виконати команду “pip install Flask” (рис. 3.1.8).

Після виконання даної команди в системі буде встановлений Flask та його залежності, що дозволить створювати веб-додатки на Python.

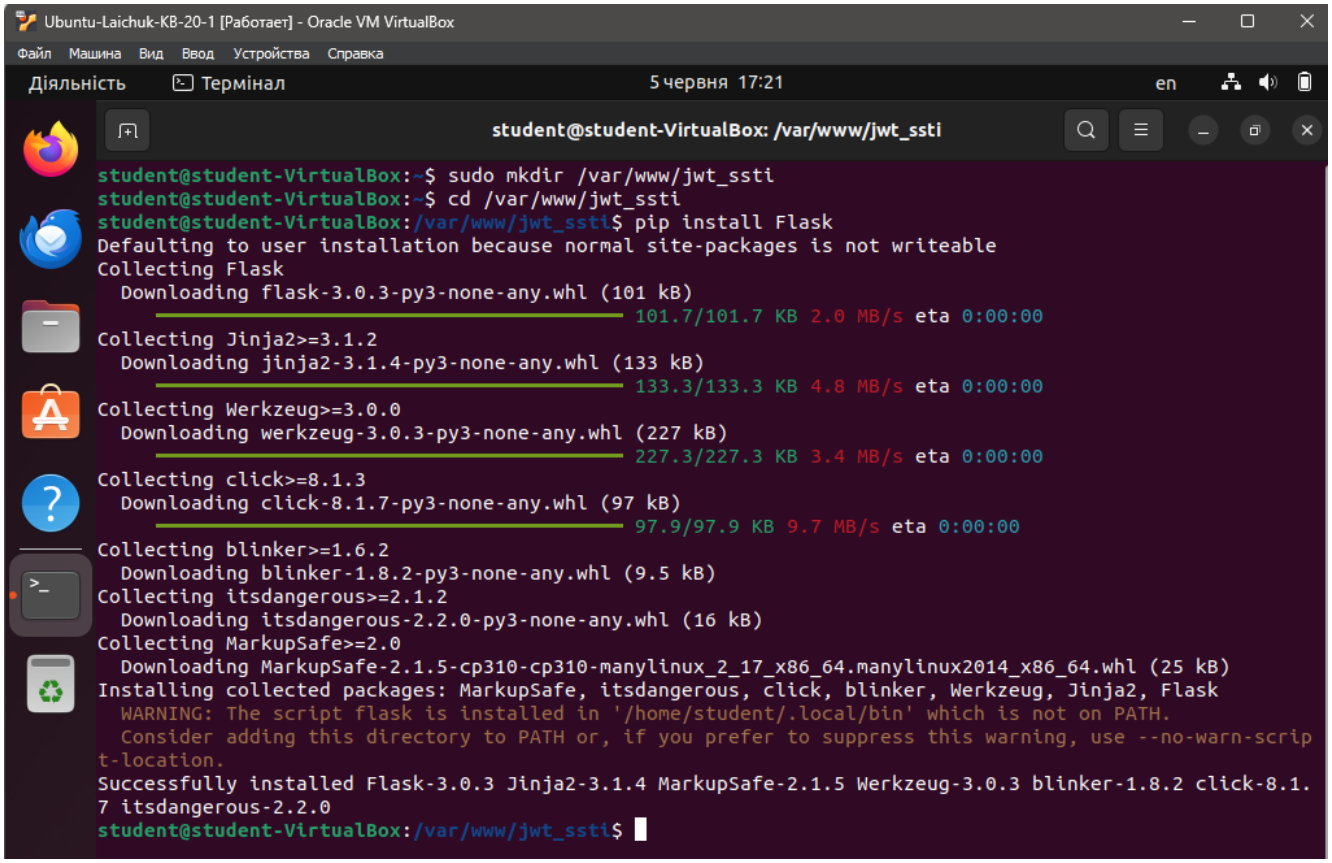


Рисунок 3.1.8 – Успішне створення каталогу та встановлення веб-фреймворка Flask

На даному етапі підготовка для розробки веб-додатків на мові програмування Python та за допомогою веб-фреймворка Flask завершена. Для розробки веб-додатків на мові PHP, слід встановити наступні компоненти:

– php - інтерпретатор мови програмування PHP, який використовується для розробки динамічних веб-сторінок і серверних скриптів;

– libapache2-mod-php - Модуль для веб-сервера Apache, який дозволяє запускати PHP-скрипти. Цей модуль інтегрує PHP з Apache, дозволяючи серверу обробляти файли PHP і виконувати PHP-код;

– php-mysql - розширення для PHP, яке забезпечує взаємодію між PHP і базою даних MySQL. Це дозволяє PHP-скриптам підключатися до MySQL-баз даних, виконувати запити, отримувати дані та виконувати інші операції з базою даних.

Встановимо вищеописані компоненти командою “sudo apt install php libapache2-mod-php php-mysql -y” (рис. 3.1.9 - 3.1.10).

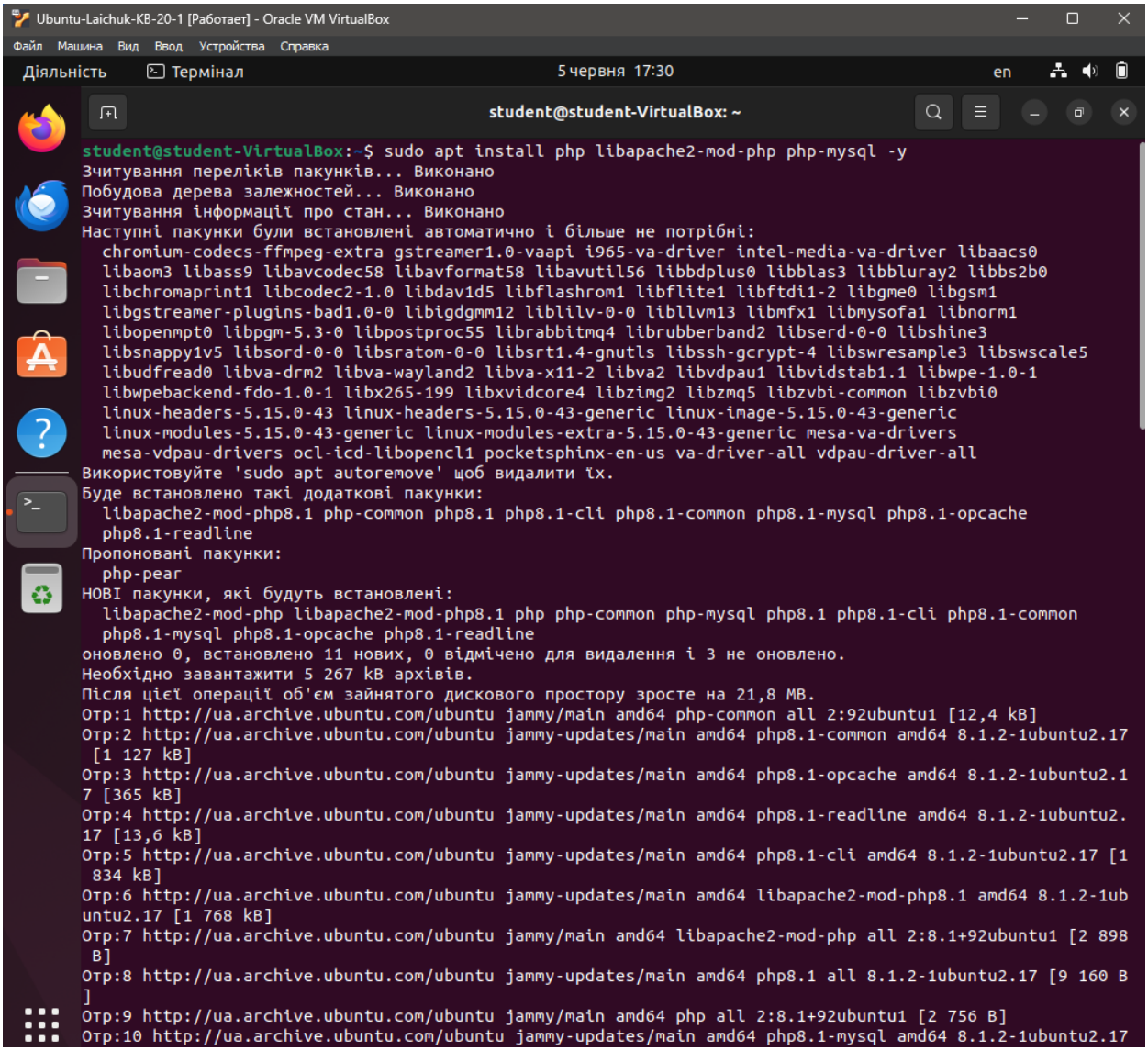


Рисунок 3.1.9 – Процес встановлення компонентів “php”, “libapache2-mod-php” та “php-mysql”

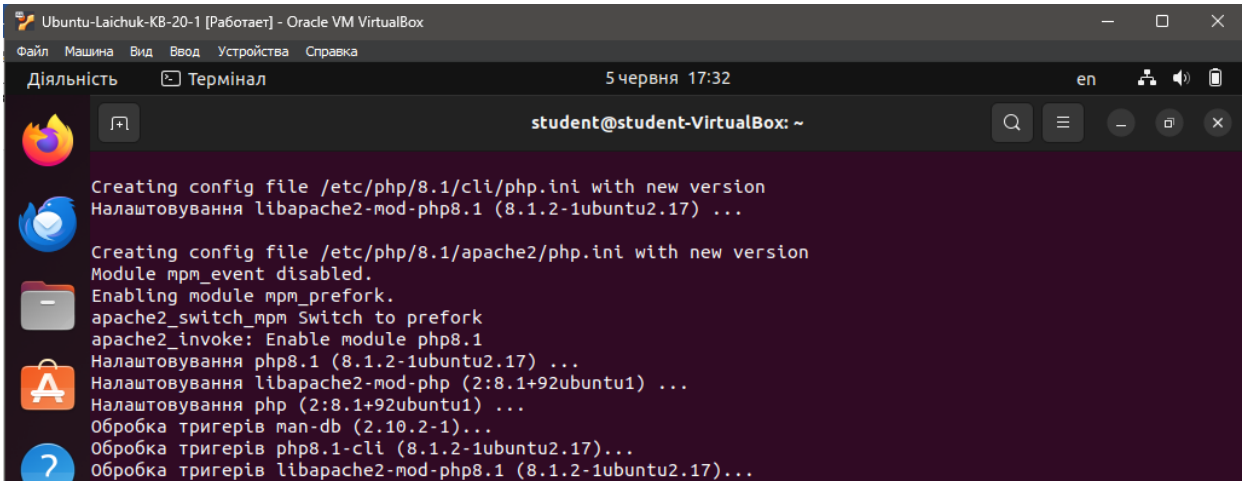


Рисунок 3.1.10 – Успішне встановлення компонентів “php”, “libapache2-mod-php” та “php-mysql”

Встановимо сервер MySQL на операційну систему “Ubuntu” за допомогою команди “sudo apt install mysql-server -y” (рис. 3.1.11 – 3.1.12).

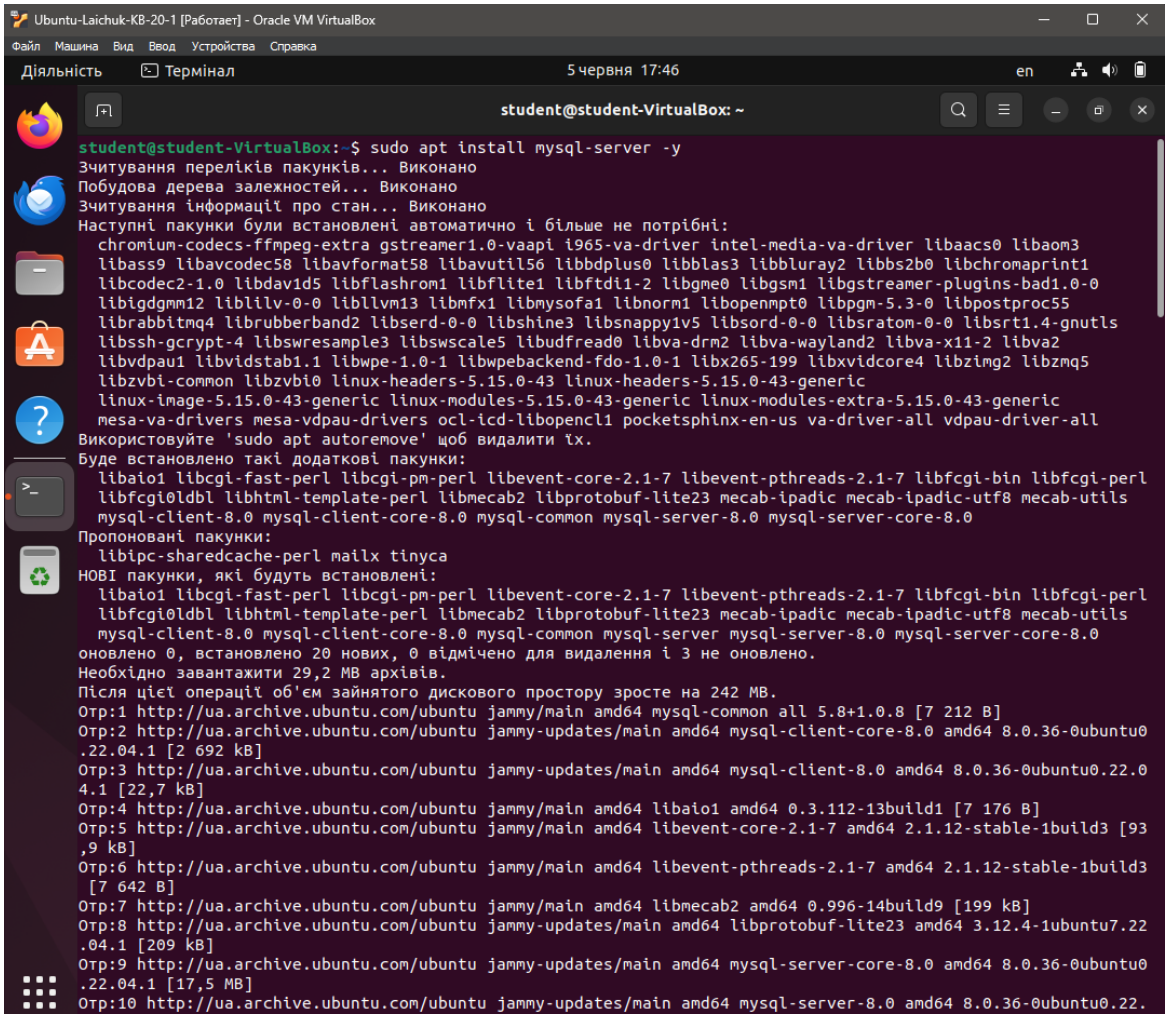


Рисунок 3.1.11 – Процес встановлення сервера MySQL

Перевіримо статус встановленого сервера за допомогою команди “sudo systemctl status mysql”.

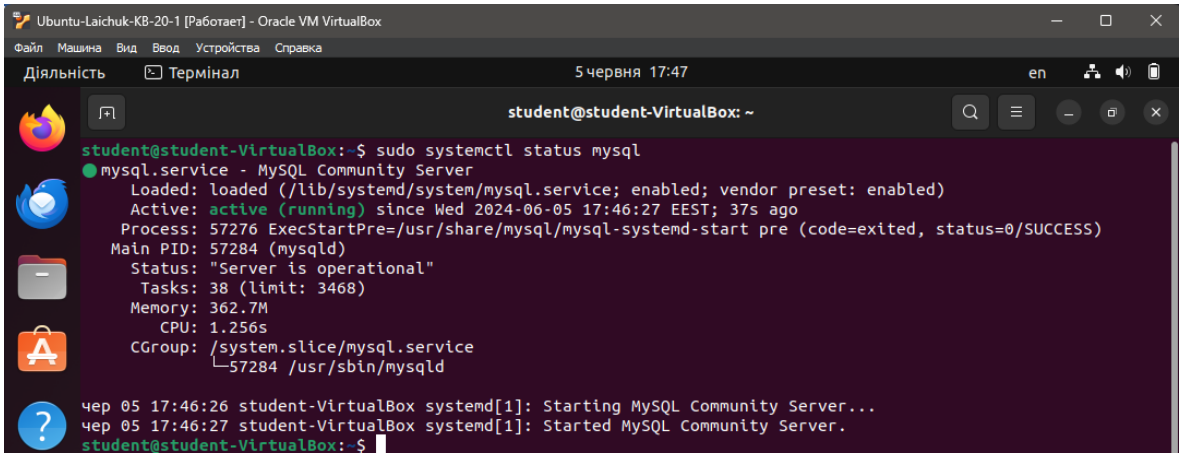


Рисунок 3.1.12 – Успішно встановлений сервер MySQL

Виконаємо також скрипт початкового налаштування безпеки сервера MySQL (рис. 3.1.13). Даний скрипт включає або виключає наступні параметри:

– Зміна паролю користувача root.

– Видалення анонімних користувачів.

– Відключення віддаленого доступу для користувача root.

– Видалення тестових баз даних і таблиць.

– Оновлення привілеїв і налаштувань.

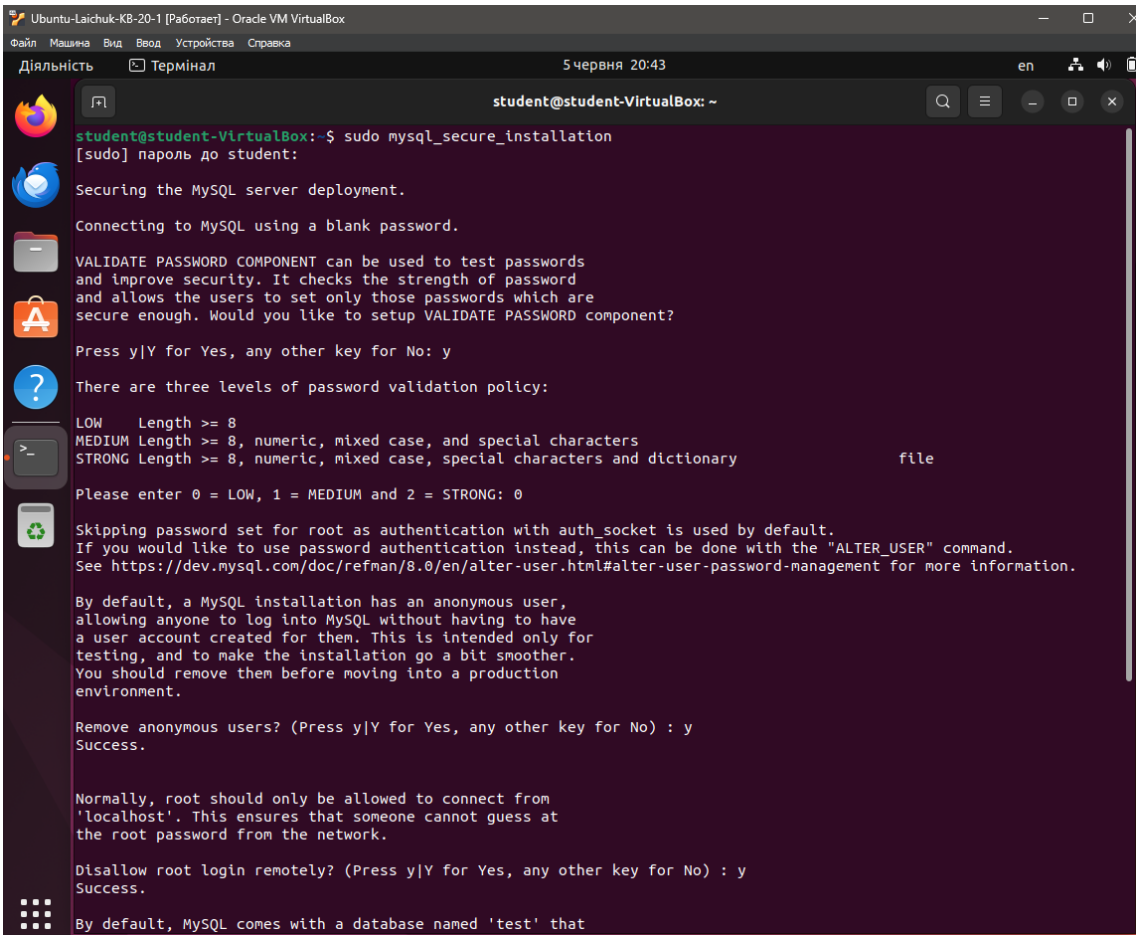


Рисунок 3.1.13 – Виконання скрипта початкового налаштування безпеки сервера MySQL

Для написання коду також встановимо редактор коду “Codium” (рис. 3.1.14). Встановлення виконується командою “sudo snap install codium --classic”. Snap-пакети є контейнеризованими додатками, які включають всі необхідні залежності, що спрощує їх установку і забезпечує більшу портативність між різними дистрибутивами Linux.

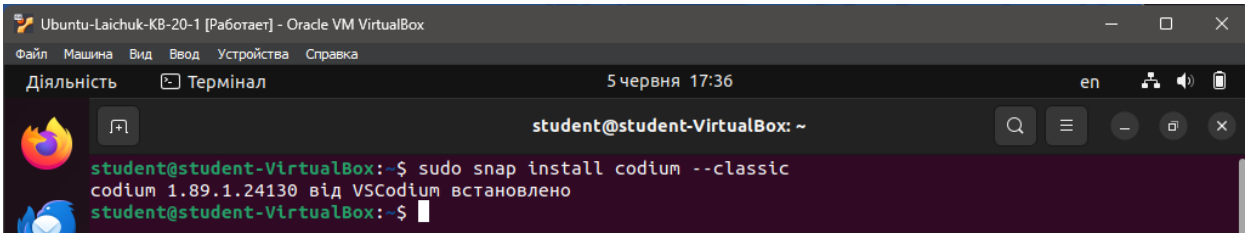


Рисунок 3.1.14 – Успішне встановлення редактора коду “Codium”

Після встановлення вищеописаних компонентів та модулів, підготовка середовища до розробки веб-сайтів вважається виконаною. Створення та редагування конфігураційних файлів відбуватиметься паралельно з розробкою веб-сайтів.

3.2. Розробка веб-сайту для дослідження пошуку прихованих шляхів та модифікації HTTP-запитів від клієнта до сервера.

3.2.1. Створення веб-сайту.

Для першого сайту було створено окремий каталог “/var/www/dirb-headers”. Всередині цього каталогу створимо файл index.php, каталог /secret, всередині якого створимо файли flag.php, .htaccess та s3cr3t_f1l3.txt

Файл index.php є головною сторінкою веб-сайту, тому додамо підказку для того, щоб учасник розумів з чого почати виконання завдання. Загальний код файлу index.php наведено на рис. 3.2.1.

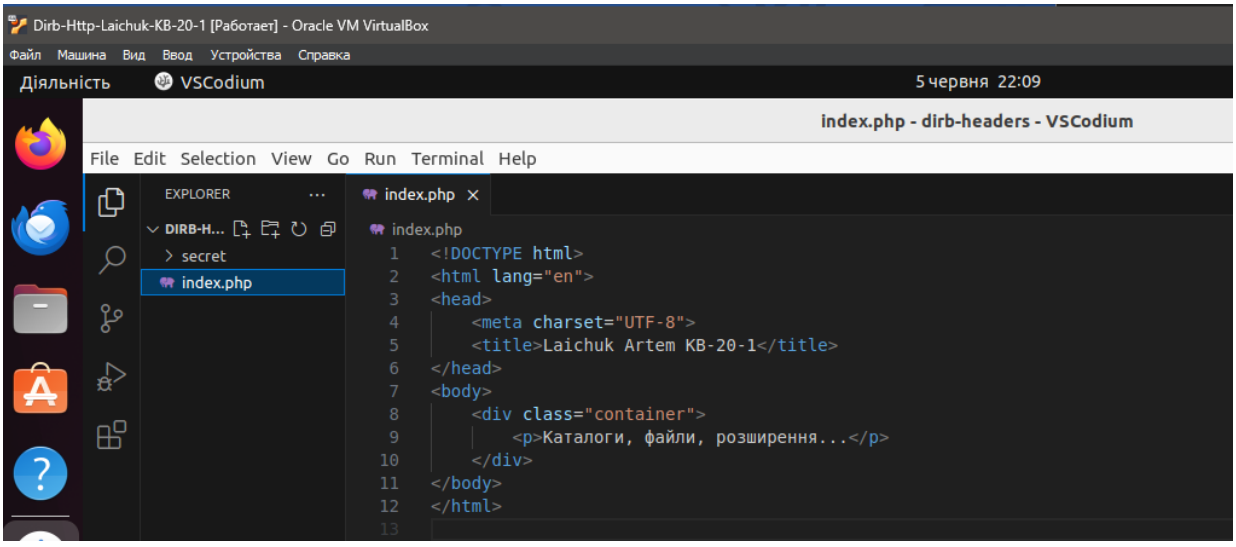


Рисунок 3.2.1 – Вміст головної сторінки index.php

Перейдемо до каталогу /secret. Файл s3cr3t_f1l3.txt зберігатиме в собі прапор, який користувач отримає в результаті успішного виконання завдання (рис. 3.2.2).

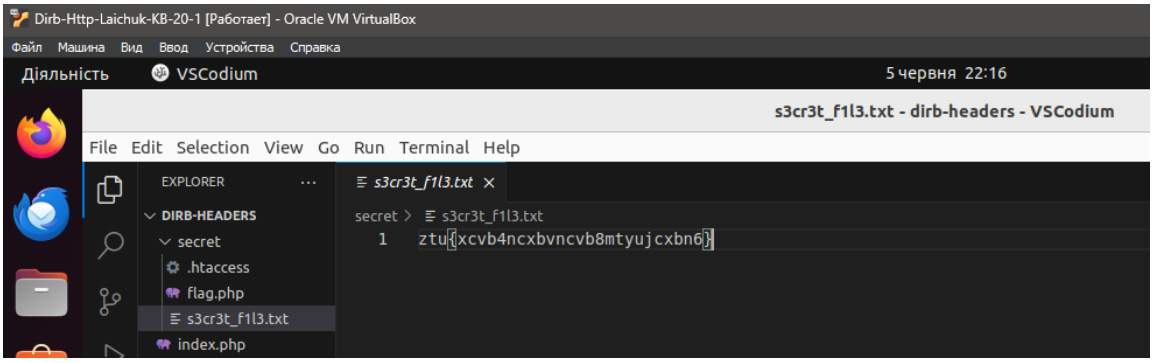


Рисунок 3.2.2 – Вміст файлу s3cr3t_f1l3.txt для зберігання прапора

Файл .htaccess необхідний для випадку, коли учасник знайде прихований каталог /secret та спробує переглянути його вміст. Додамо до цього файлу рядок “Options -Indexes” який встановить заборону на відображення списку файлів та захистить вміст директорії від перегляду користувачами, які можуть намагатися знайти конфіденційні або важливі файли.

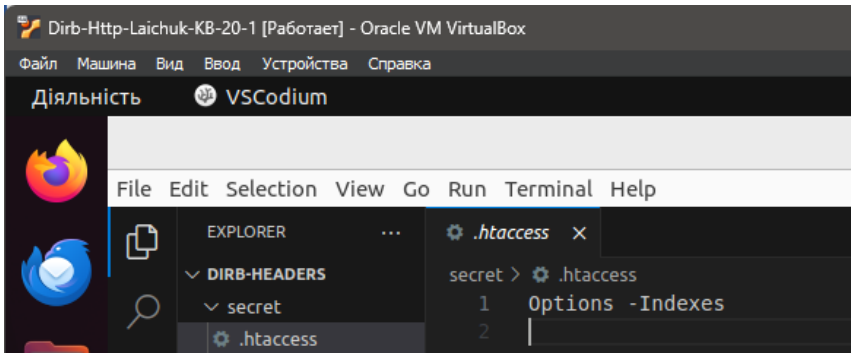


Рисунок 3.2.3 – Вміст файлу .htaccess

Головним файлом є flag.php, адже саме в ньому необхідно реалізувати функції для перевірки HTTP-запитів, що надходять на сервер. Для захоплення прапора, учасник повинен буде модифікувати або додати такі поля як:

– Referer - заголовок вказує адресу сторінки, з якої було здійснено перехід до запитуваного ресурсу.

– User-Agent - заголовок містить інформацію про клієнтський додаток, який здійснює запит.

– X-Forwarded-For - заголовок використовується для передачі оригінальної IP-адреси клієнта через проксі-сервер або балансувальник навантаження.

– Date - заголовок вказує дату і час, коли було створено запит. Формат відповідає стандарту RFC 1123.

Також додамо такий заголовок як “Range” для того, щоб користувачу одразу не відображався весь прапор, а лише його частина. Необхідно буде додати цей заголовок та встановити відповідне значення довжини прапора.

Для початку створимо асоціативний масив з назвою “$required_headers”, який міститиме перші чотири вищеописані заголовки HTTP-запиту.

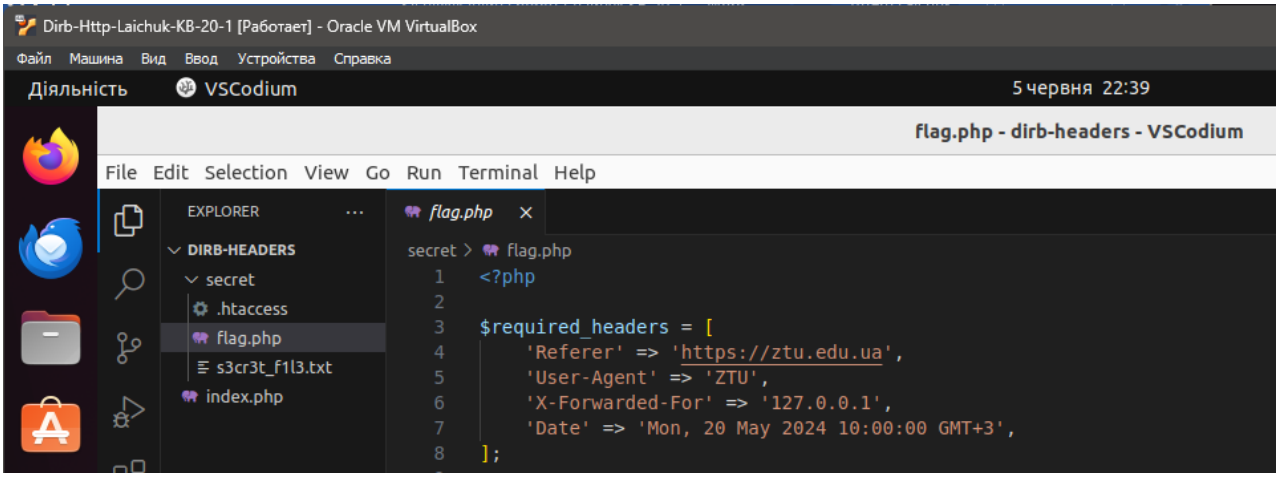


Рисунок 3.2.4 – Вміст масиву “$required_headers” із заголовками HTTP-запиту

В даному фрагменті коду, кожен елемент масиву представляє собою пару “ключ-значення”, де ключ – це назва заголовка, а значення – відповідне значення цього заголовка. Таким чином, заголовок “Referer” повинен містити в собі посилання на офіційний сайт Державного університету “Житомирська політехніка” – https://ztu.edu.ua, заголовок “User-Agent” повинен мати значення “ZTU”, заголовок “X-Forwarded-For” – значення “127.0.0.1” для імітації переходу до веб-сайта з використанням локального хосту, а заголовок “Date” – значення “Mon, 20 May 2024 10:00:00 GMT+3” для встановлення конкретної дати та часу при надсиланні HTTP-запиту до веб-сервера.

Також додамо умовну логіку для обробки HTTP-запиту на основі наявності та значення заголовка “HTTP_RANGE”. Наступний фрагмент коду (рис. 3.2.5) відповідає за читання вмісту файлу з прапором, часткову його відправку учаснику або виведення помилки про неповний дівпазон, що встановлений користувачем.

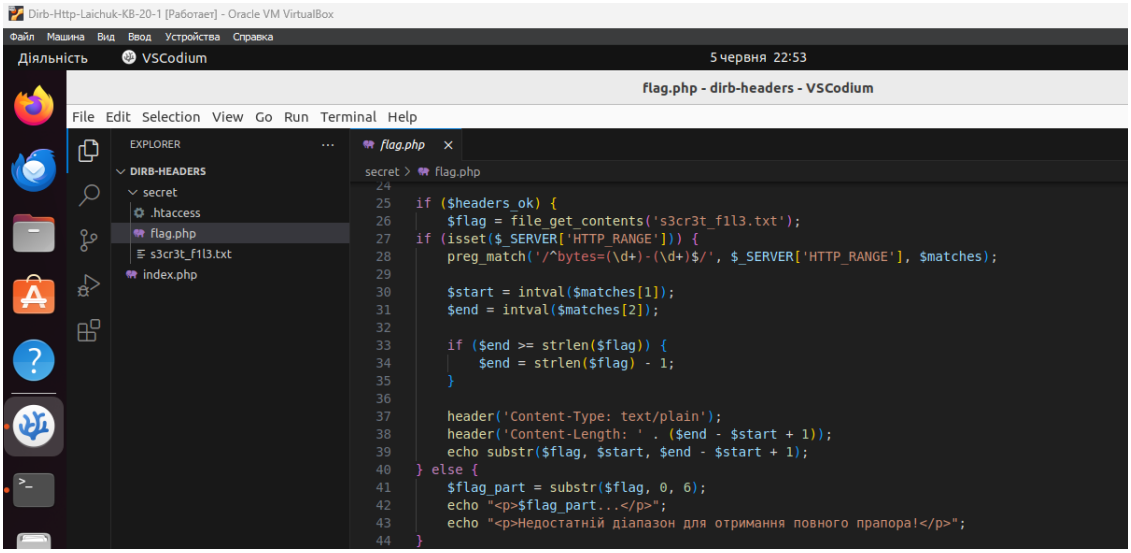


Рисунок 3.2.6 – Логіка для обробки HTTP-запиту на основі наявності та значення заголовка “HTTP_RANGE”

Для кожного заголовку додамо підказку, щоб учасник розумів, який саме заголовок потрібно модифікувати або додати (рис. 3.2.7).

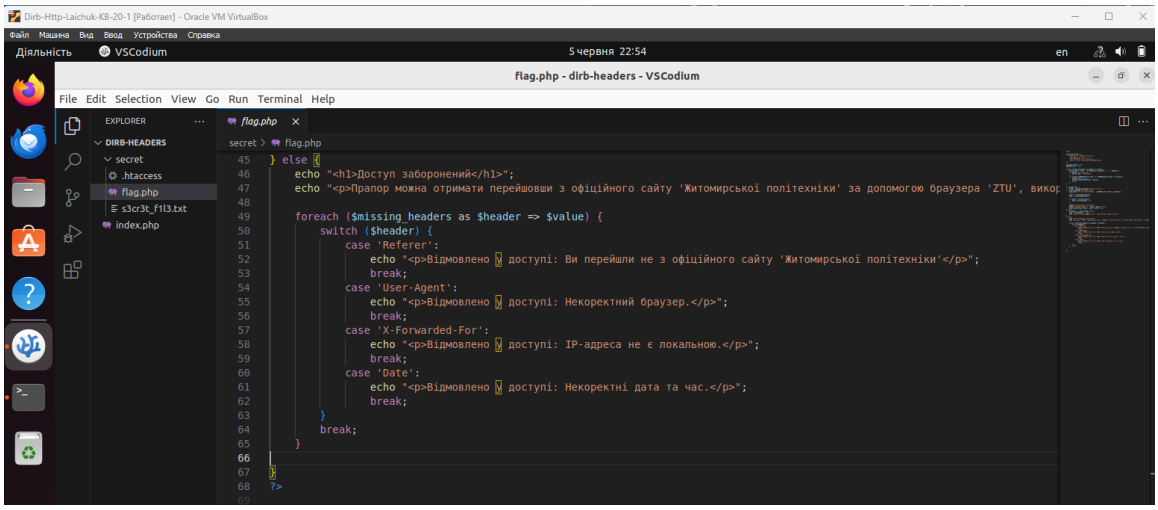


Рисунок 3.2.7 – Додавання підказок для кожного заголовка HTTP-запиту

Для розгортання даного веб-сайту за допомогою веб-сервера Apache, необхідно створити відповідний конфігураційний файл за шляхом: “ /etc/apache2/sites-available/”. Загальна структура та вміст конфігураційного файлу dirb-headers.conf наведені на рис. 3.2.8.

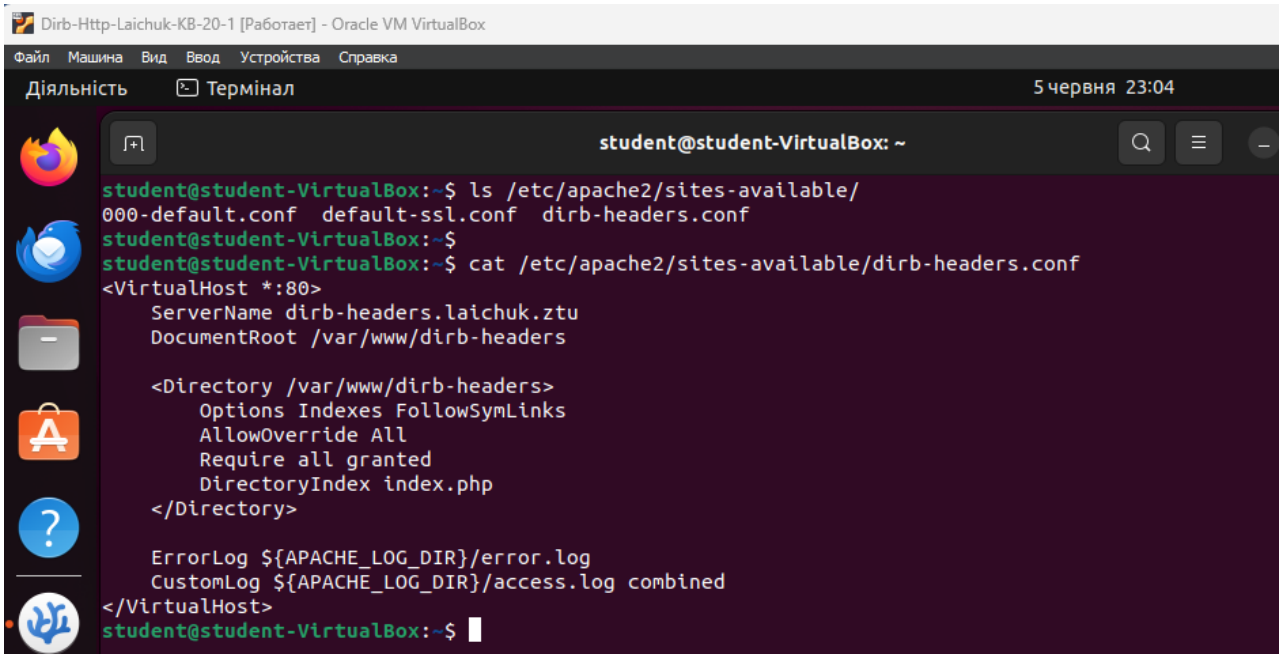


Рисунок 3.2.8 – Створений конфігураційний файл та його вміст для налаштування віртуального хоста веб-сервері Apache

Загалом, цей файл конфігурації налаштовує веб-сервер Apache для обслуговування веб-сайту dirb-headers.laichuk.ztu, який знаходиться в директорії /var/www/dirb-headers. Він налаштовує дозволи доступу, обробку індексних файлів, дозволяє використання .htaccess файлів і визначає місця для журналів помилок і доступу.

Останнім кроком є запуск веб-сервера Apache (рис. 3.2.9) командою “sudo systemctl start apache2”, після чого можна розпочати тестування веб-сайту та виконання дій для захоплення прапора.

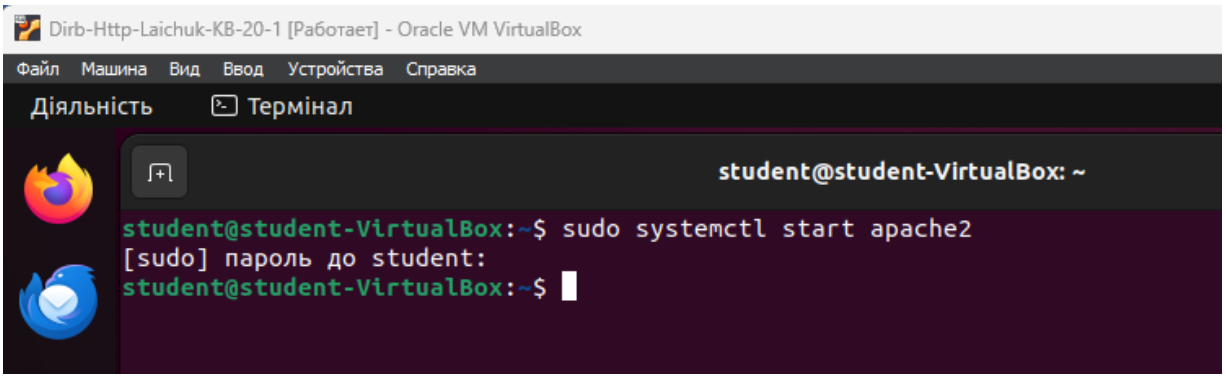


Рисунок 3.2.9 – Успішний запуск веб-сервера Apache

3.2.2. Тестування розробленого функціоналу та його експлуатація для захоплення прапору.

Для зв’язку між робочою станцією учасника та веб-сервером в середовищі VirtualBox створимо мережу NAT з назвою “Laichuk-KB-20-1”, IP-адресою мережі 201.9.10.0 (де 9 – номер групи, а 10 – номер варіанта) та маскою /24 (255.255.255.0).

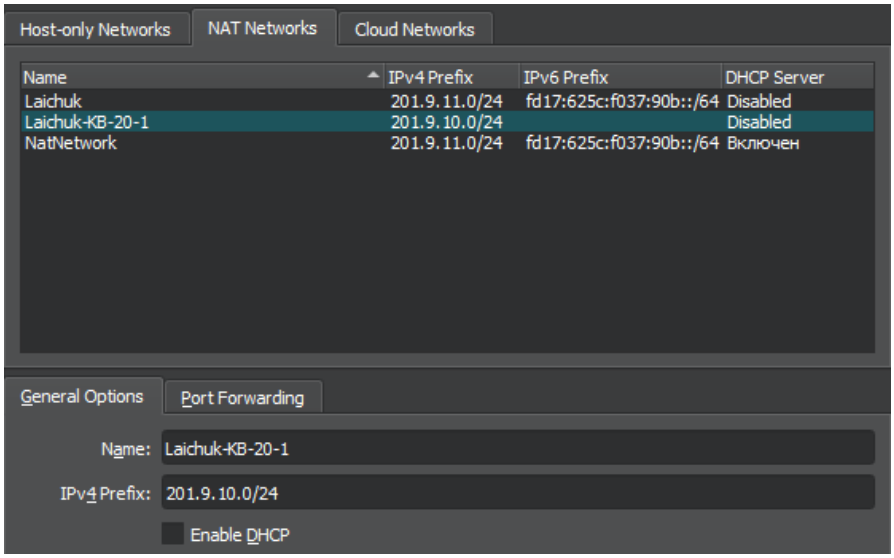


Рисунок 3.2.10 – Створена мережа NAT для зв’язку між робочою станцією учасника

та веб-серверами

Веб-серверу присвоїмо IP-адресу 201.9.10.5/24, а робочій станції учасника – 201.9.10.2/24. Перевіримо інформаційний обмін між робочою станцією та веб-сервером.

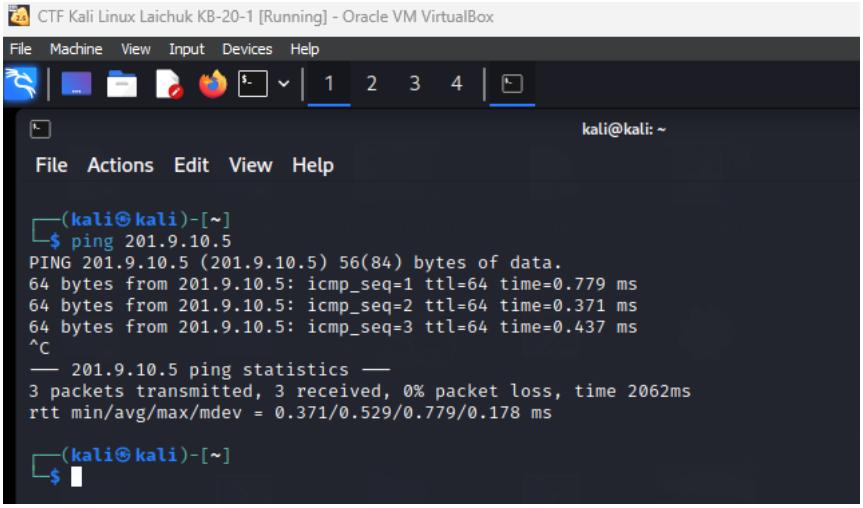


Рисунок 3.2.11 – Успішний інформаційний обмін між робочою станцією

та веб-сервером

Додамо до файлу /etc/hosts IP-адресу веб-сервера та ім’я, за яким будемо звертатись до веб-сервера за допомогою браузера.

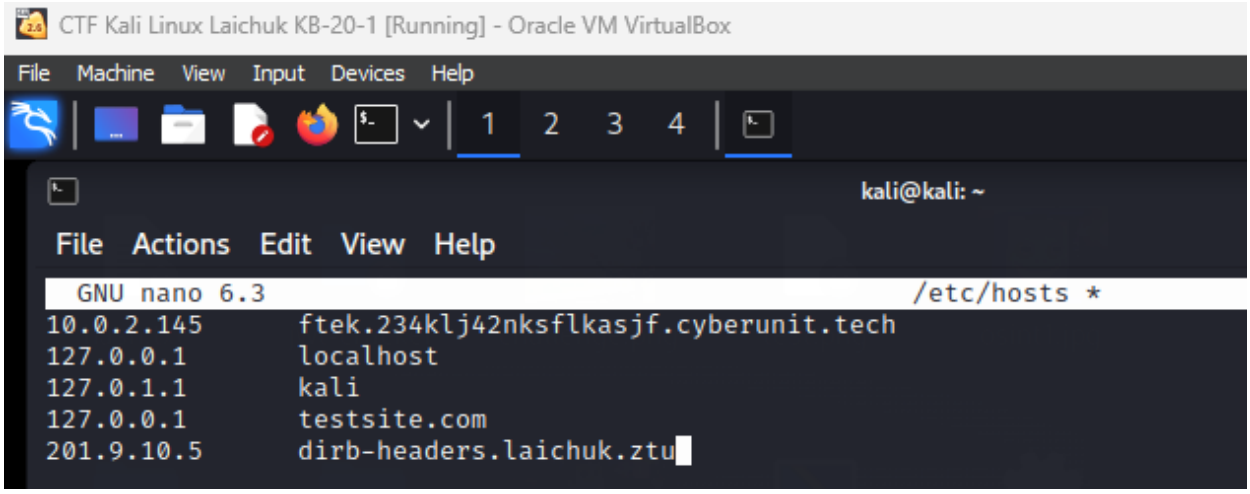


Рисунок 3.2.12 – Успішно додана IP-адреса та доменне ім’я веб-сервера до файлу /etc/hosts на робочій станції Kali Linux

Тепер за допомогою браузера та доменного ім’я веб-сервера спробуємо відкрити розроблений веб-сайт.

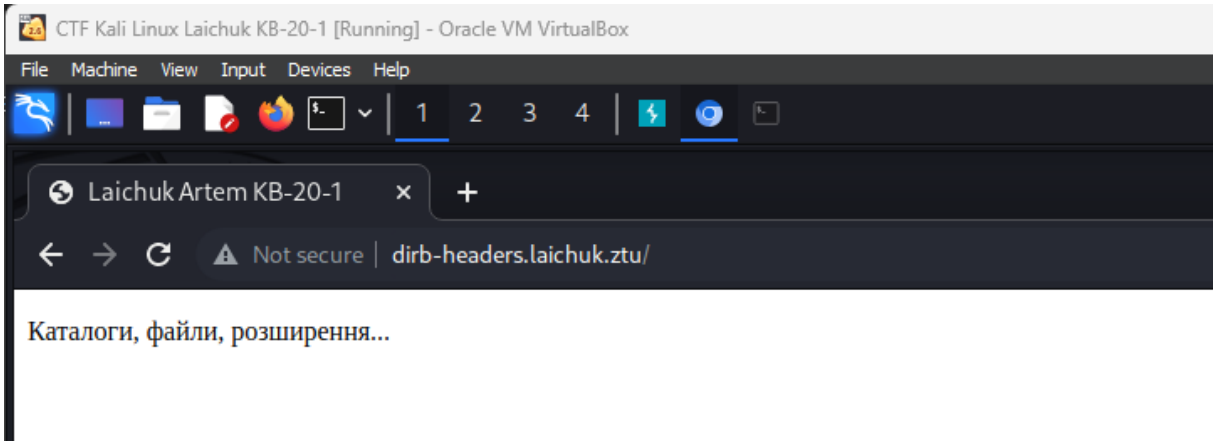


Рисунок 3.2.13 – Успішне відкриття веб-сайту за допомогою браузера

Розпочнемо процес захоплення прапора. Першим кроком необхідно виконати пошук прихованих каталогів та файлів. Скористаємось інструментом “dirb” та словником за замовчуванням.

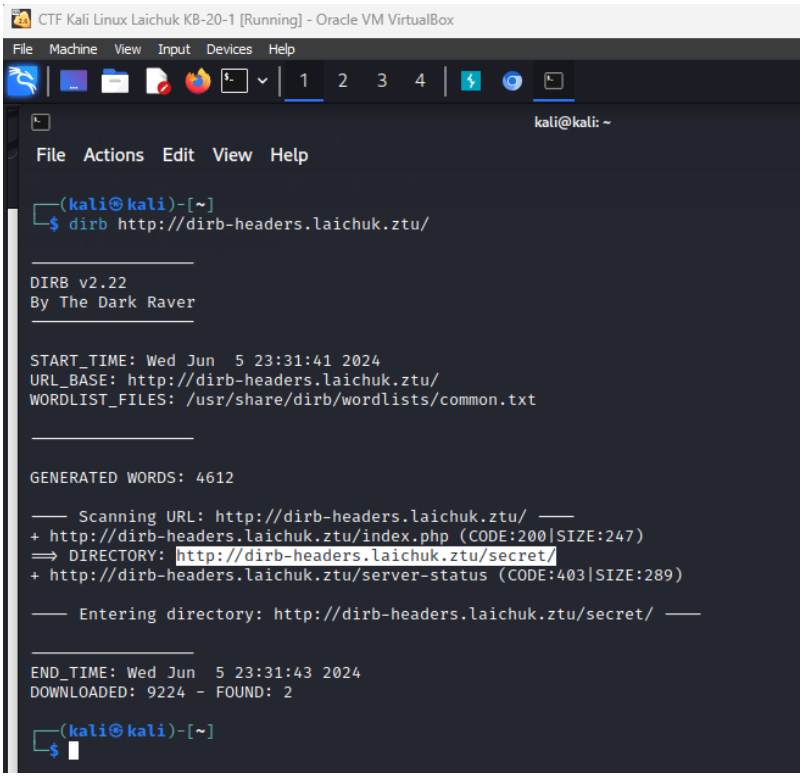


Рисунок 3.2.14 – Успішно знайдений прихований каталог за допомогою інструмента dirb та атаки за словником

Продовжимо пошуки у знайденому каталозі /secret з використанням параметра для встановлення типу файлів для пошуку, таких як txt, php тощо. Для цього використаємо параметр -X.

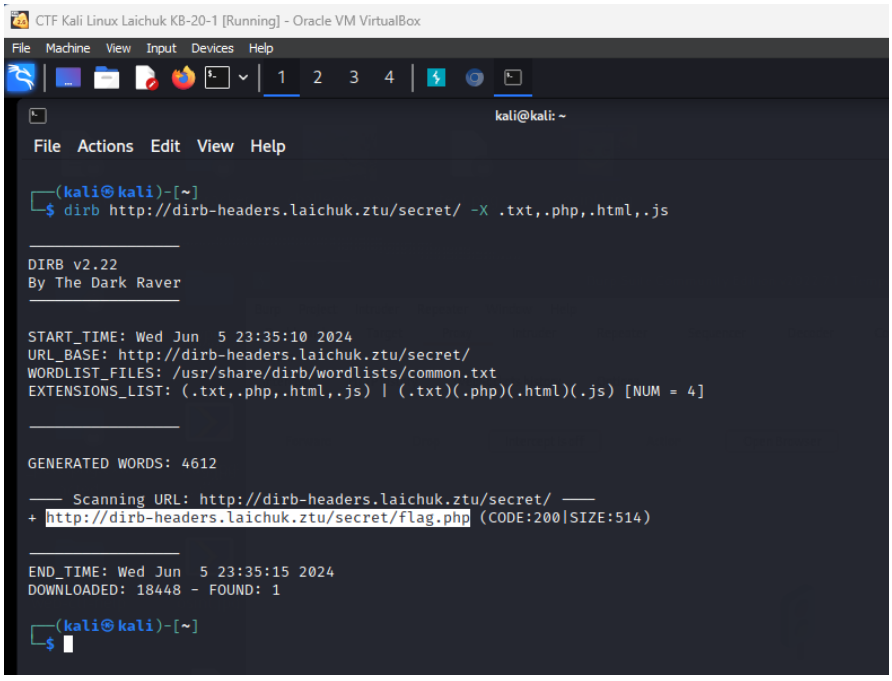


Рисунок 3.2.15 – Успішно знайдений файл flag.php у каталозі /secret за допомогою інструмента dirb та атаки за словником

Перейдемо за шляхом /secret/flag.php на веб-сайті.

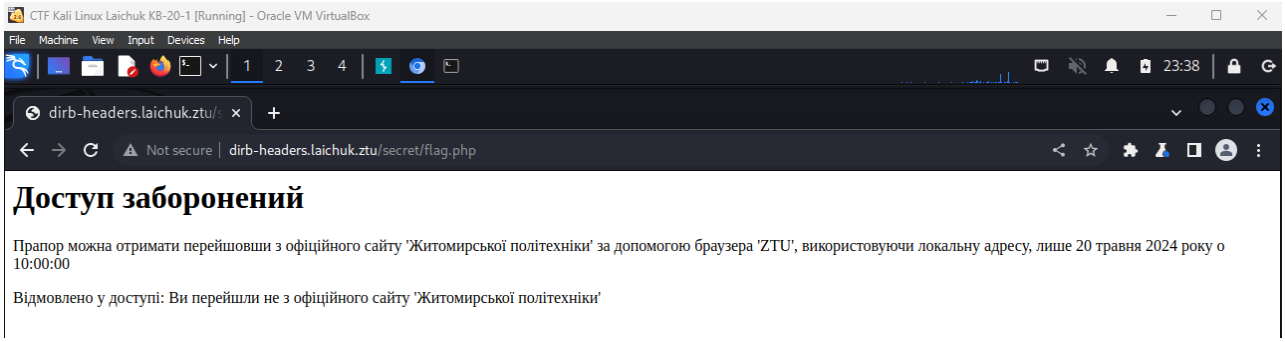


Рисунок 3.2.16 – Успішний перехід за шляхом /secret/flag.php

Як бачимо на рис. 3.2.16, перед нами одразу з’явилася підказка, в якій говориться, що перейти за ших шляхом необхідно з офіційного сайту ДУ “Житомирська політехніка”. Для модифікації HTTP-запитів використаємо інструмент “BurpSuite”. Згідно підказки, перехопимо HTTP-запит від клієнта до сервера та додамо відповідний заголовок. Виконаємо аналогічні дії для всіх наступних підказок (рис. 3.2.17 – 3.2.).

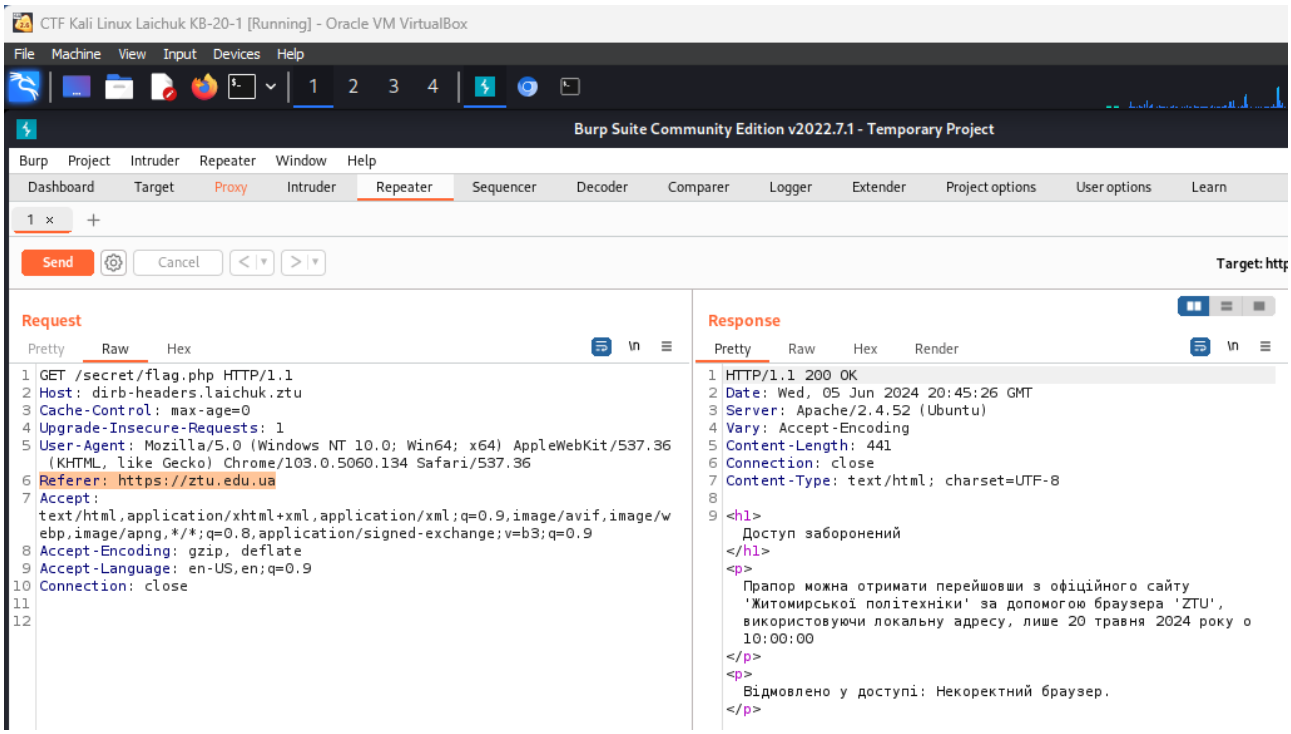


Рисунок 3.2.17 – Успішне перехоплення HTTP-запиту, додавання заголовка “Referer” та отримання наступної підказки

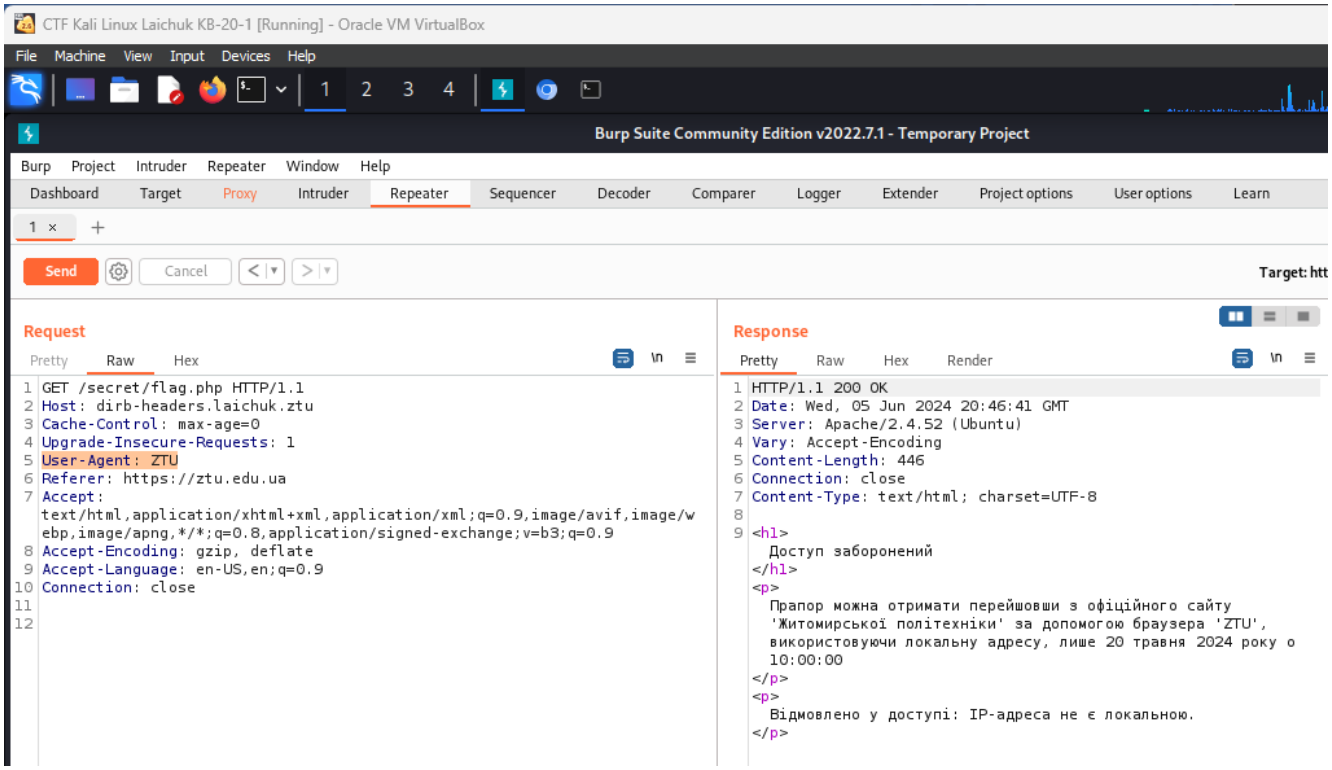


Рисунок 3.2.18 – Успішна зміна заголовка “User-Agent” та отримання наступної підказки

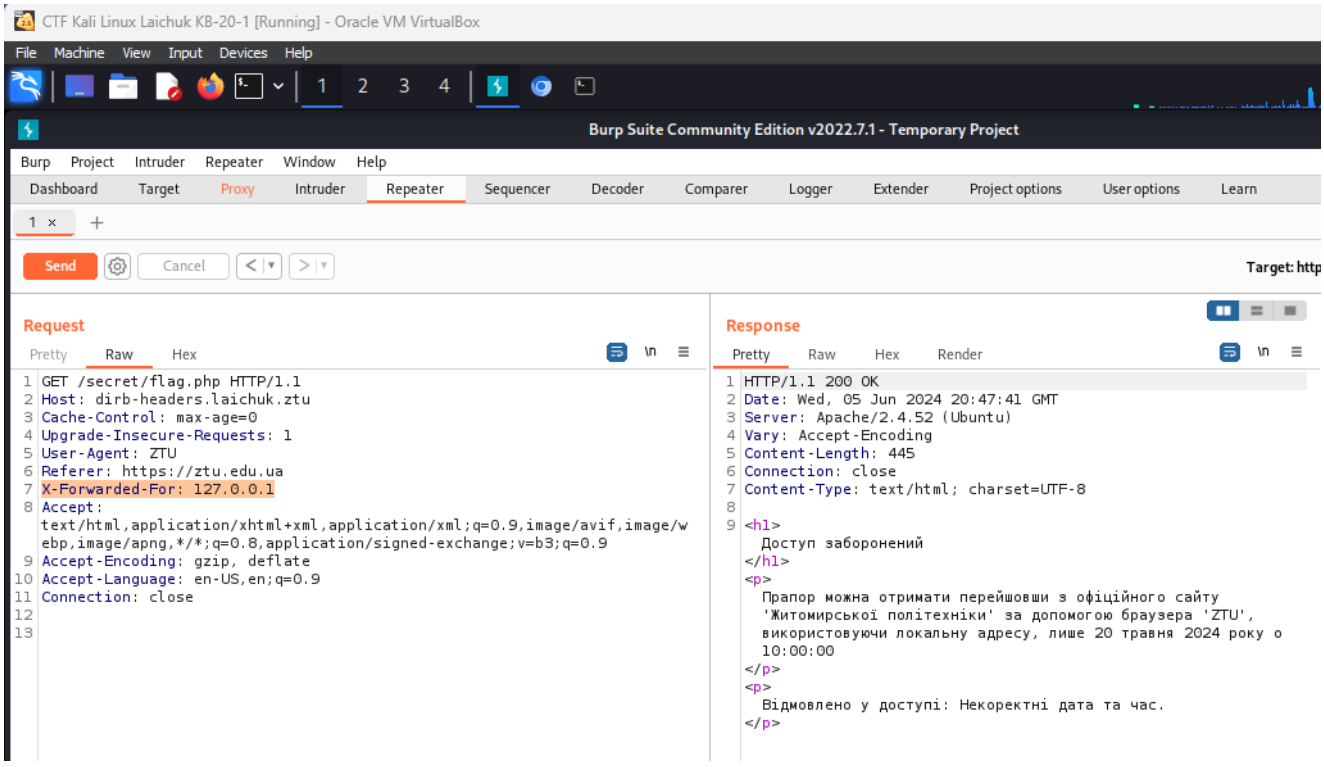


Рисунок 3.2.19 – Додавання заголовка “X-Forwarded-For” до HTTP-запиту та успішне отримання наступної підказки

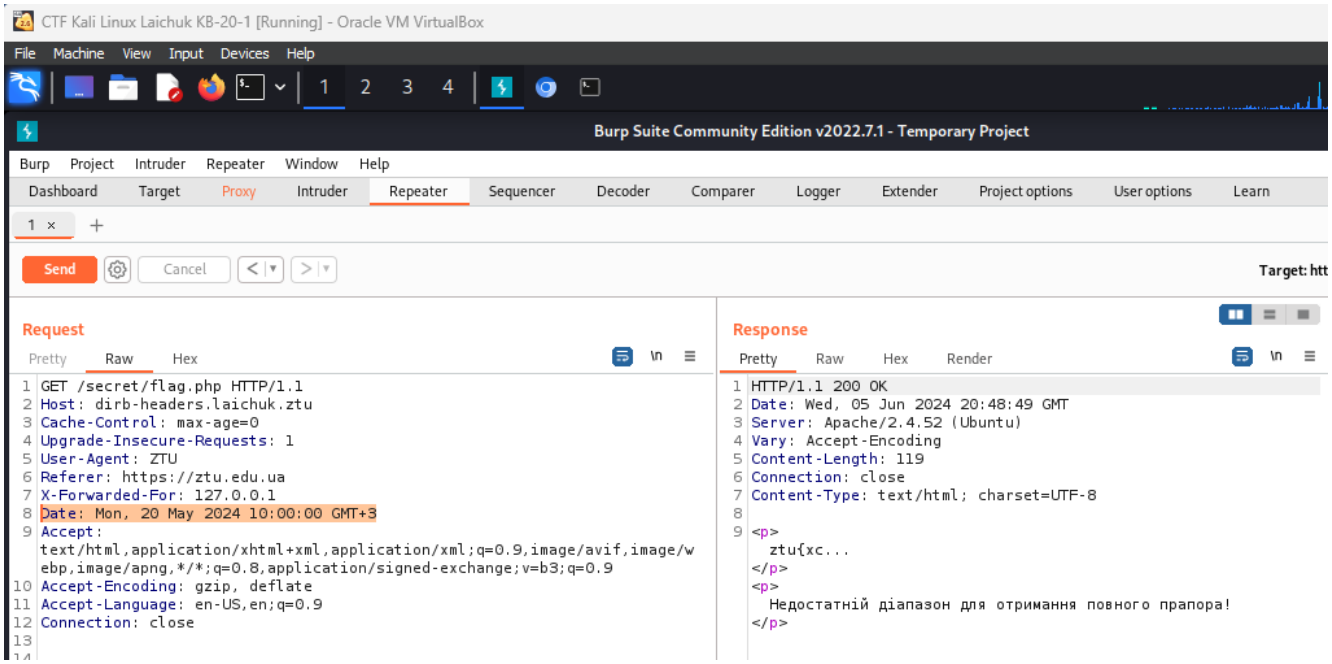


Рисунок 3.2.20 – Додавання заголовка “Date” з потрібним значенням до HTTP-запиту та успішне отримання частини прапора

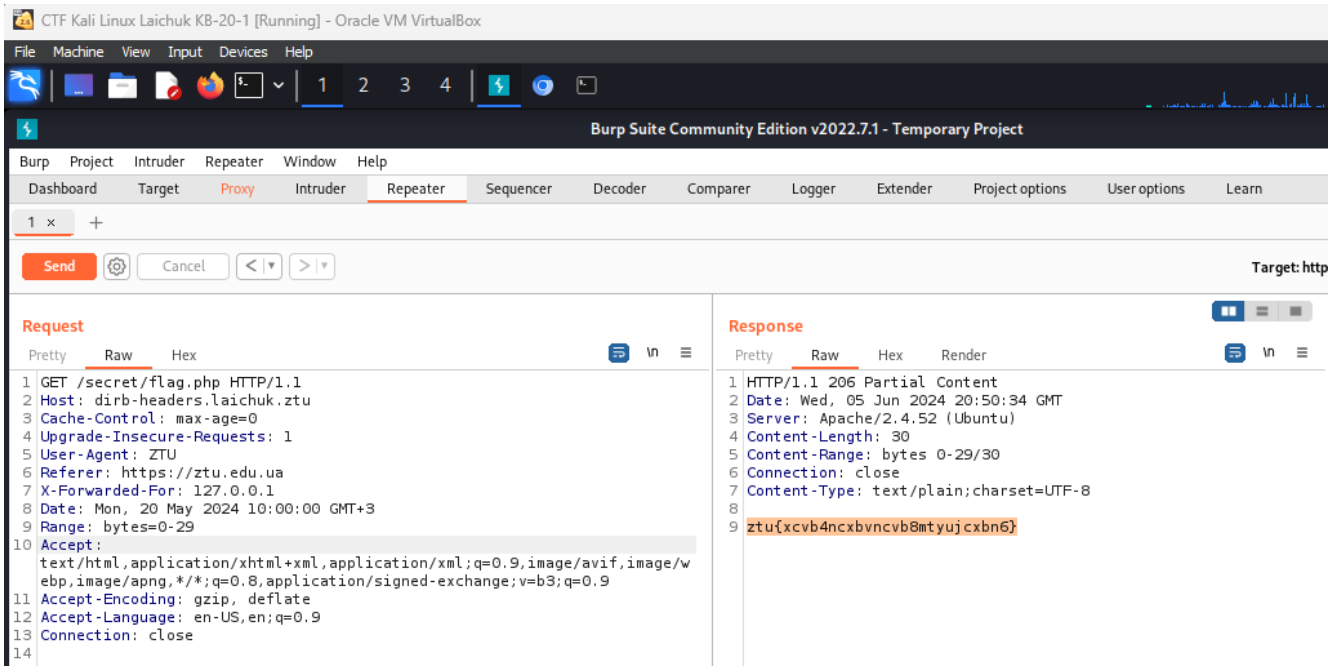


Рисунок 3.2.21 – Додавання заголовка “Range” із значенням “bytes=0-29” до HTTP-запиту та успішне отримання повного прапора

Після виконання вищеописаних дій, прапор був захоплений, що свідчить про успіше виконання завдання №1.

3.3. Розробка веб-додатку з вразливостями JSON Web Token та Server-Side Template Injection та їх експуатація.

3.3.1. Створення веб-сайту.

Загальна файлова структура проекту виглядає наступним чином:

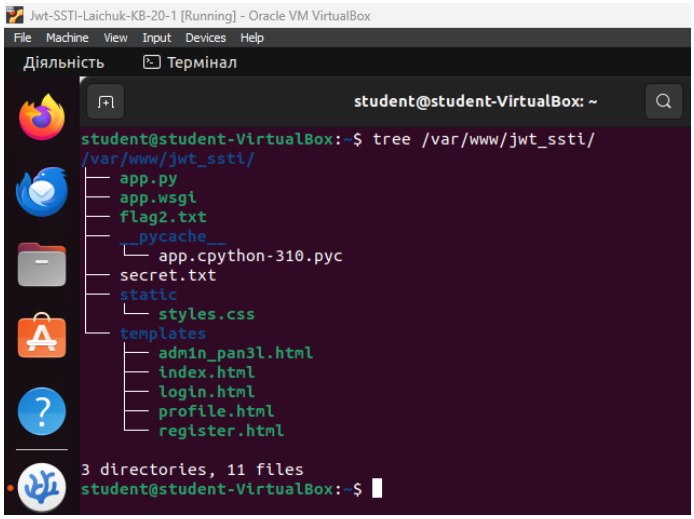


Рисунок 3.3.1 – Файлова структура проекту

Розглянемо найважливіші фрагменти коду, де будуть реалізовані вразливості. Весь код проекту буде винесено в додатки пояснювальної записки.

Перша вразливість – використання слабкого секретного ключа для токена JWT. В даній роботій таким ключем буде ‘i1U2We3THEY4!’. Також одразу створимо словник, який буде зберігати дані про облікові записи користувачів (логін, пароль та роль облікового запису), в якому буде захована підказка (рис. 3.3.2).

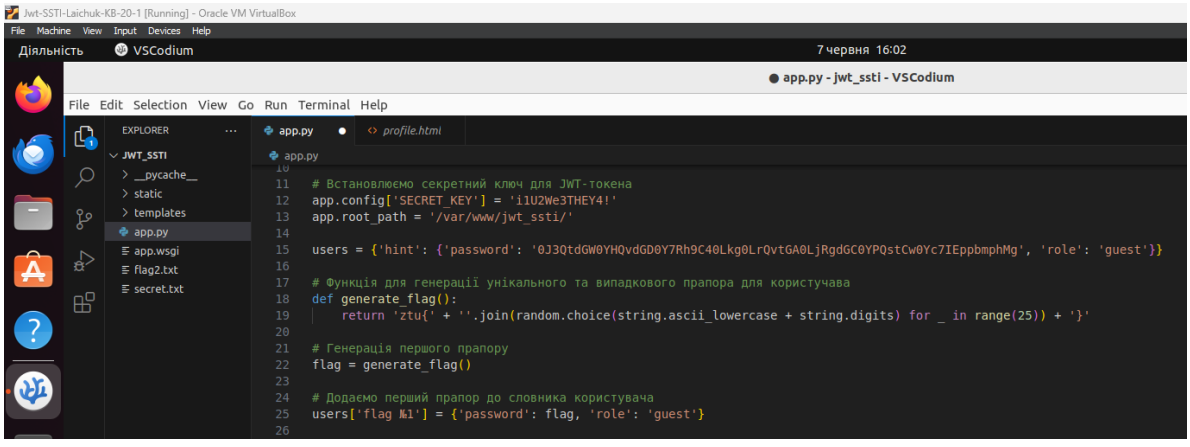


Рисунок 3.3.2 – Встановлення слабкого секретного ключа для токена JWT, створення словника для зберігання облікових даних користувачів та занесення до нього першого згенерованого прапора

Розглянемо також процес формування токена JWT (рис. 3.3.3).

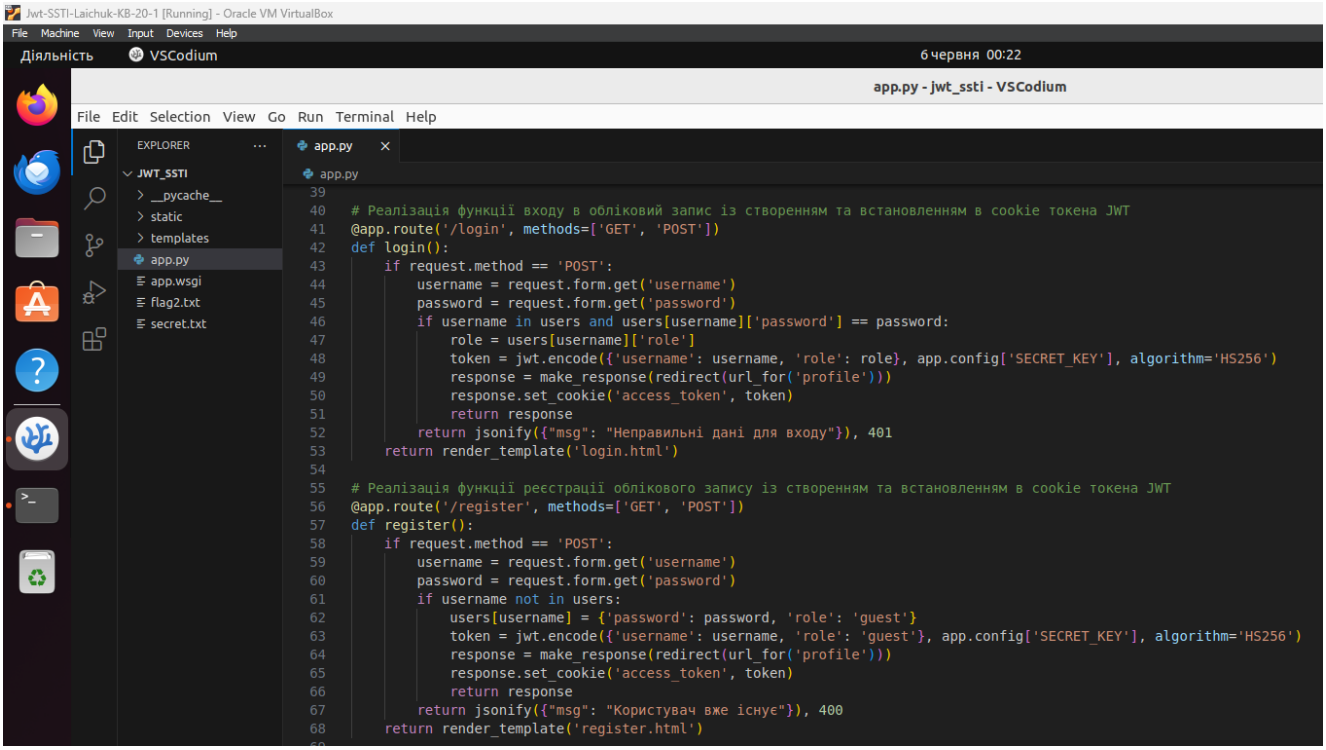


Рисунок 3.3.3 – Формування JWT-токена у випадку успішної реєстрації та входу в обліковий запис

Формування токена JWT, згідно представленого фрагмента коду на рис. 3.3.3, відбувається наступним чином:

– У разі успішної реєстраціїї та входу в обліковий запис, за допомогою функції “jwt.encode()” з бібліотеки PyJWT формується токен JWT.

– Після створення токена, він записується в cookie access_token за допомогою методу response.set_cookie(). Cookie є частиною HTTP-відповіді і передається браузеру, який зберігає її для подальшого використання.

– Відповідь на запит формується з перенаправленням на сторінку профілю користувача за допомогою redirect(url_for('profile')).

Перший прапор можна отримати лише у випадку, якщо учасник отримає роль “admin” для свого облікового запису та перейде до панелі адміністратора на вебсайті, де прапор знаходитиметься в таблиці з усіма користувачами.

Щоб отримати другий прапор, учаснику необхідно буде впровадити ін’єкції Server-Side Template та за допомогою створеного корисного навантаження відобразити вміст текстового файлу, що містить прапор.

Щоб реалізувати таку вразливість, використаємо функцію ‘render_template_string’ використовується дл ярендерингу HTML-шаблону з використанням рядка, який може містити шаблонні вирази Jinja2 (рис. 3.3.4).

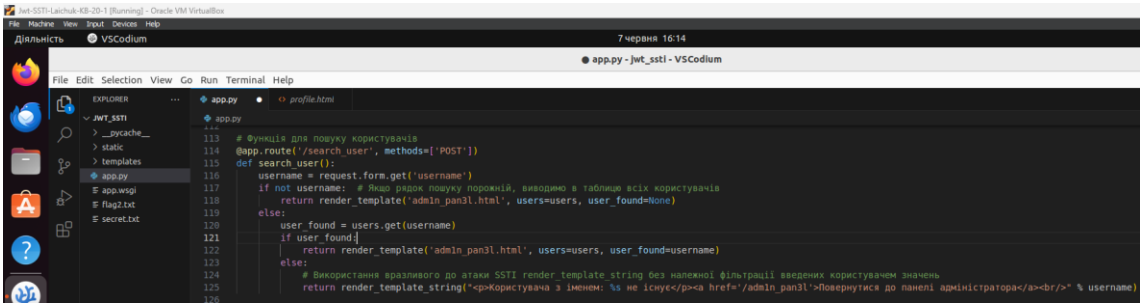


Рисунок 3.3.4 – Використання вразливої до SSTI-атак функції ‘render_template_string’

В даному випадку, значення змінної “username”, яке вводить користувач, вставляється безпосередньо в рядок шаблону. Якщо зловмисник введе спеціально сформований шаблонний вираз, він може виконати довільний код на сервері. Наприклад, якщо користувач введе {{7 * 7}}, результатом буде виведення 49 в рендереному HTML, що демонструє можливість виконання виразів.

Фінальним етапом є створення віртуального хоста та спеціального файлу з розширенням .wsgi для того, що запустити веб-сайт на сервері Apache [22] (рис. 3.3.5 - 3.3.6).

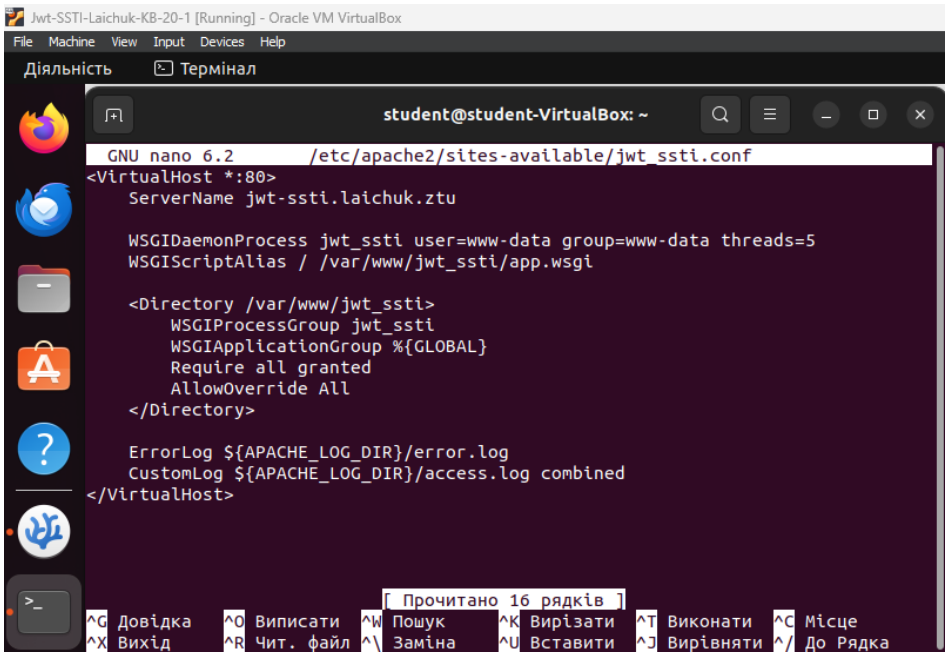


Рисунок 3.3.5 – Успішно створений конфігураційний файл для віртуального хоста

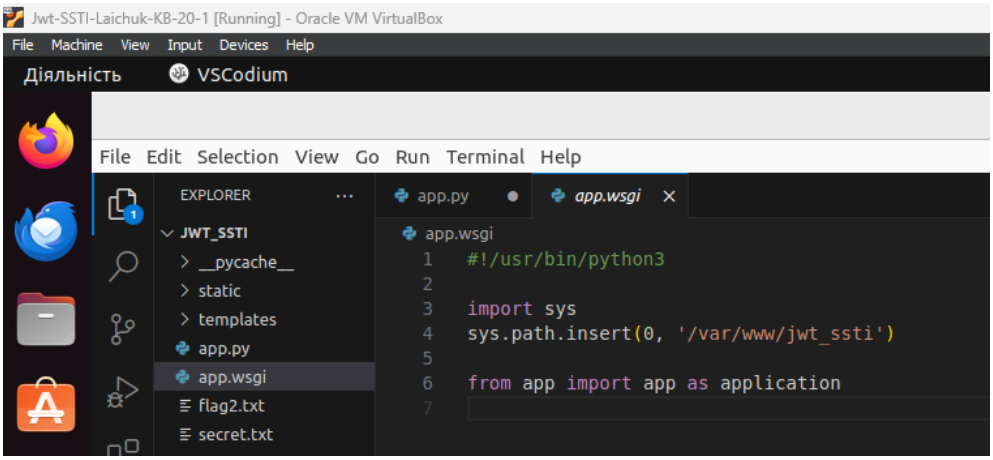


Рисунок 3.3.6 – Створений файл app.wsgi для інтеграції веб-додатку з веб-сервером Apache

Після реалізації вразливостей та описання важливих фрагментів коду перейдемо до тестування веб-сайту та експлуатації вразливостей.

3.3.2. Тестування розробленого функціоналу та експлуатація реалізованих вразливостей.

Для початку, виконаємо тестування основного функціоналу (реєстрація, вхід до облікового запису, формування токена JWT і вихід з облікового запису).

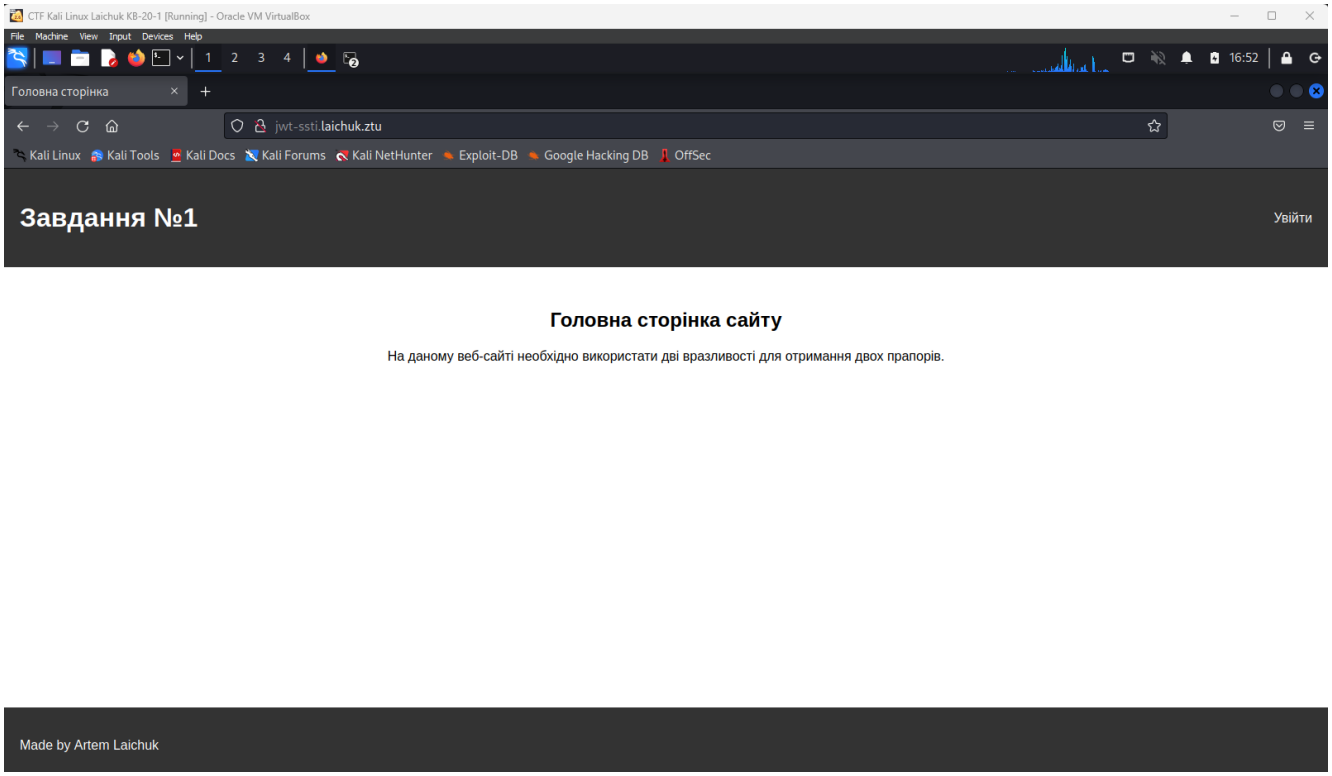


Рисунок 3.3.7 – Головна сторінка веб-сайту

Спробуємо зареєструвати обліковий запис з логіном “laichuk_artem” та аналогічним паролем.

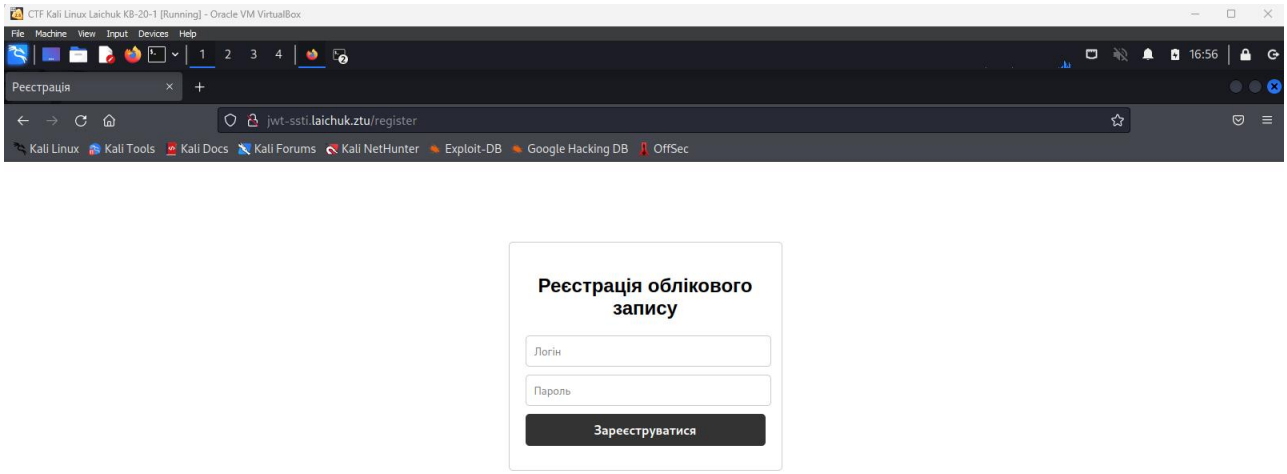


Рисунок 3.3.8 – Сторінка реєстрації облікового запису

Після успішної реєстрації облікового запису, переходимо до сторінки профілю облікового запису користувача (рис. 3.3.9).

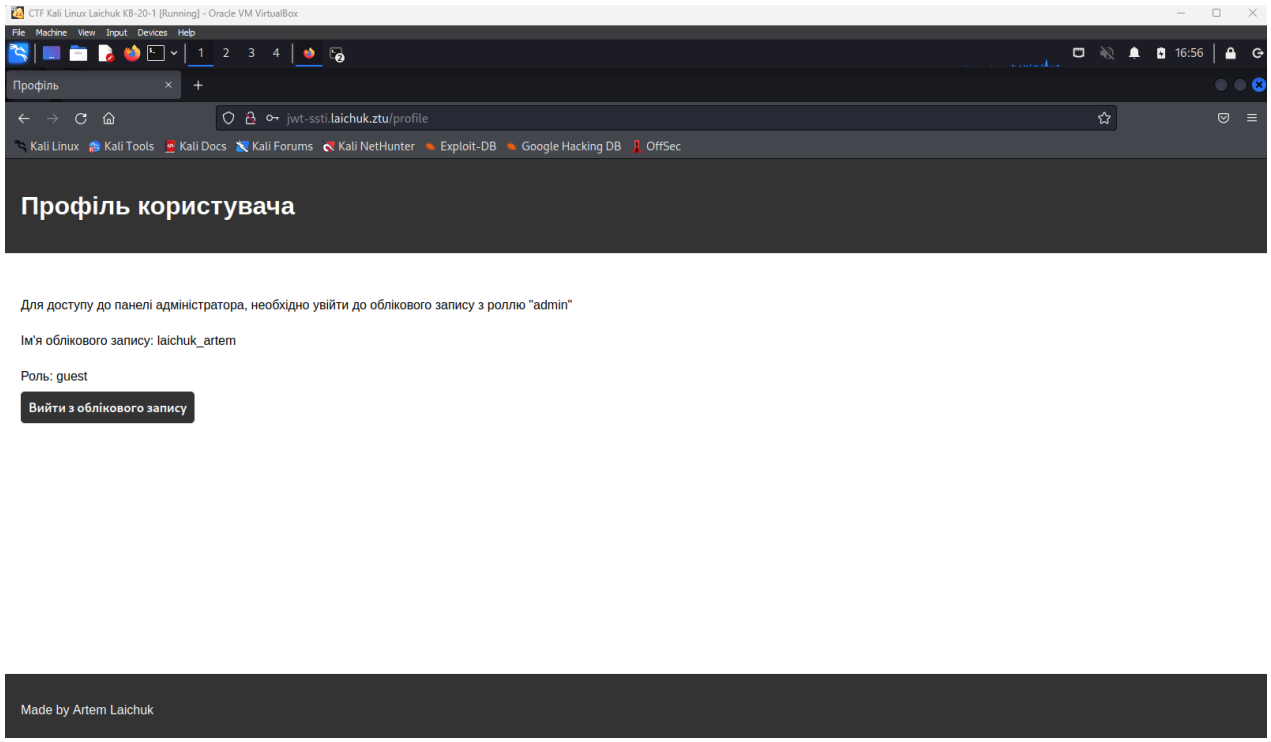


Рисунок 3.3.9 – Сторінка профілю користувача

Як бачимо на рис. 3.3.9, учаснику відображається перша підказка, ім’я та роль облікового запису, яка наразі має значення “guest”. За допомогою інструмента “Inspect” в браузері перевіримо наявність cookie та токена JWT (рис. 3.3.10).

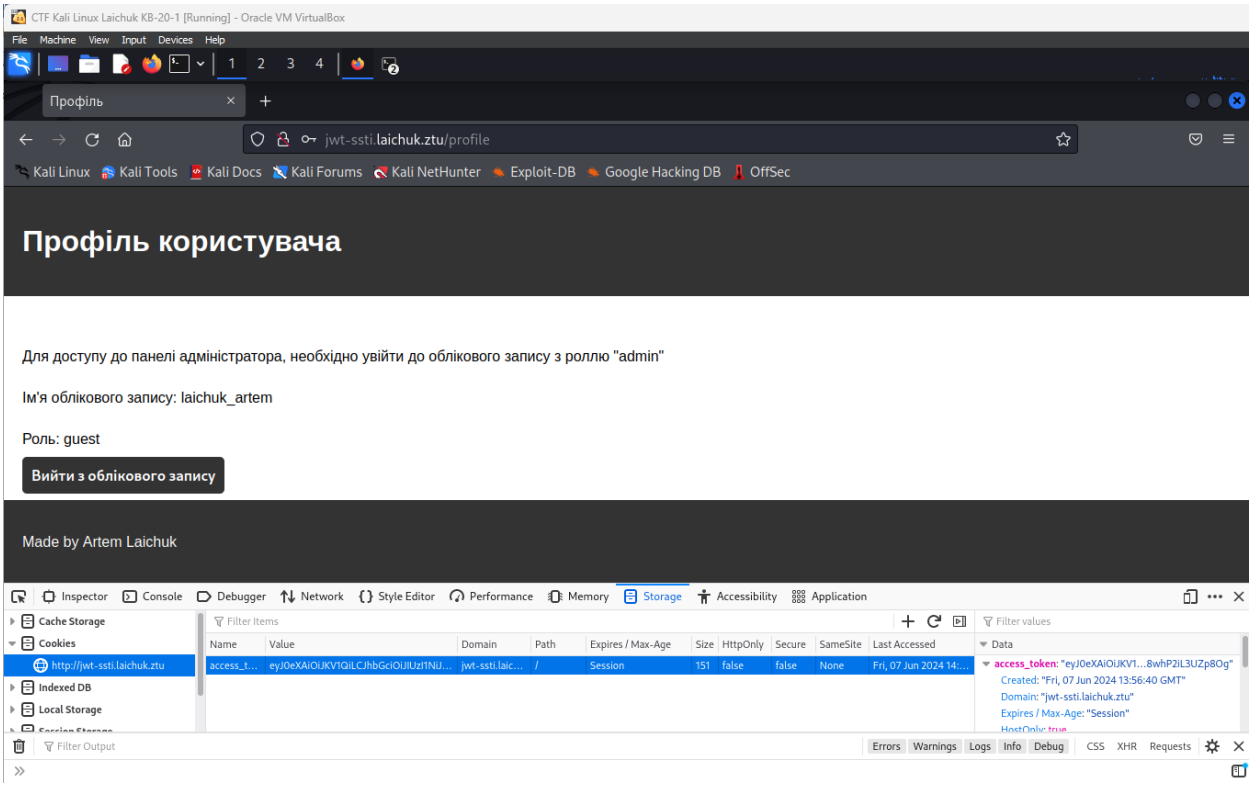


Рисунок 3.3.10 – Успішно згенерований та занесений в cookie токен JWT

Декодуємо токен за допомогою методу Base64 та перевіримо коректність даних в згенерованому токені. Необхідно розкодувати лише перших дві частини токена, що розділені крапкою, адже третя відповідає за підпис (signature) та містить випадковий набір значень.

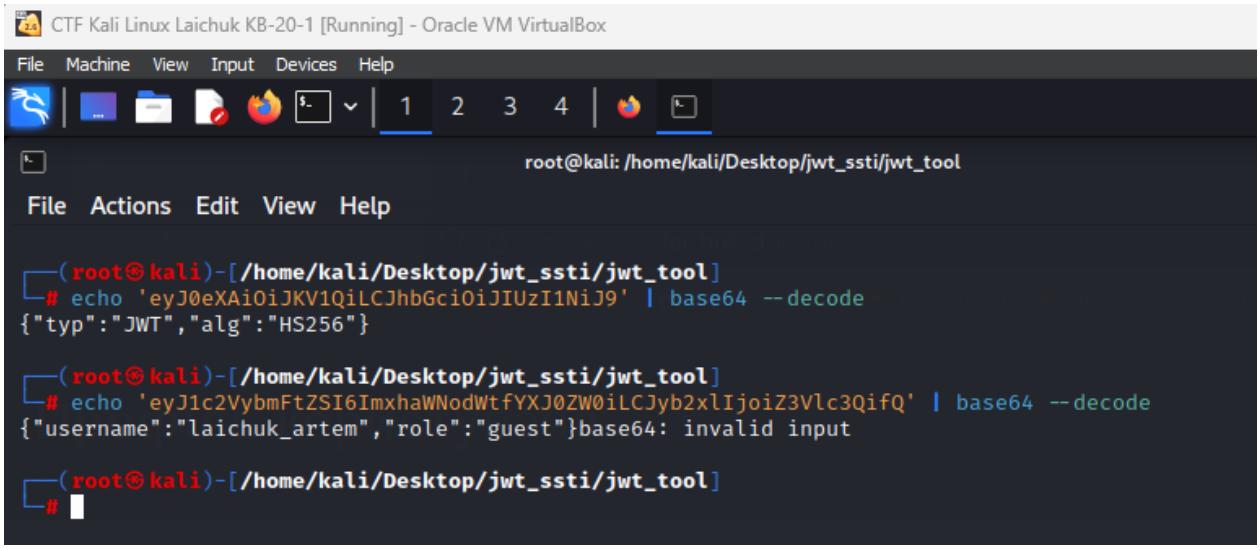


Рисунок 3.3.11 – Декодування заголовка (header) та тіла (payload) токена

Як бачимо, токе згенерований коректно та містить в собі поля з іменем облікового запису та роллю користувача.

Перейдемо до процесу отримання прапорів. Для початку, використавши інструмент “jwt_tool” (https://github.com/ticarpi/jwt_tool), здійснимо атаку за словником на секретний ключ токена JWT.

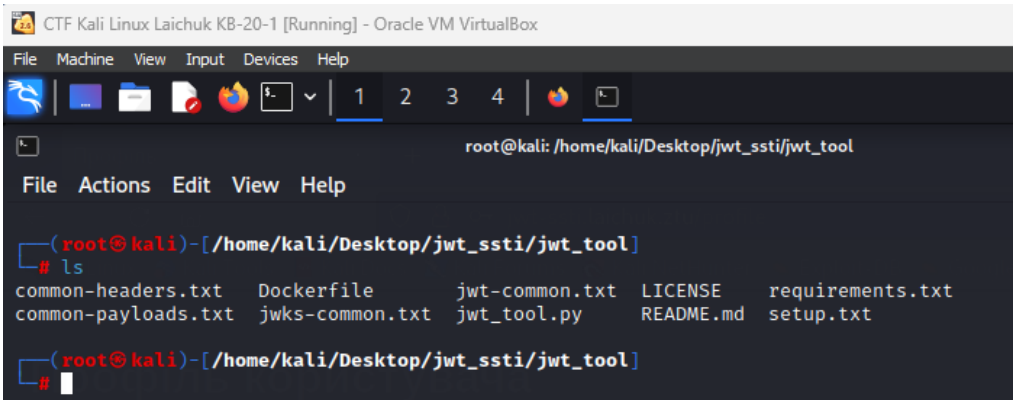


Рисунок 3.3.12 – Вміст каталогу jwt_tool для виконання атак на токен JWT

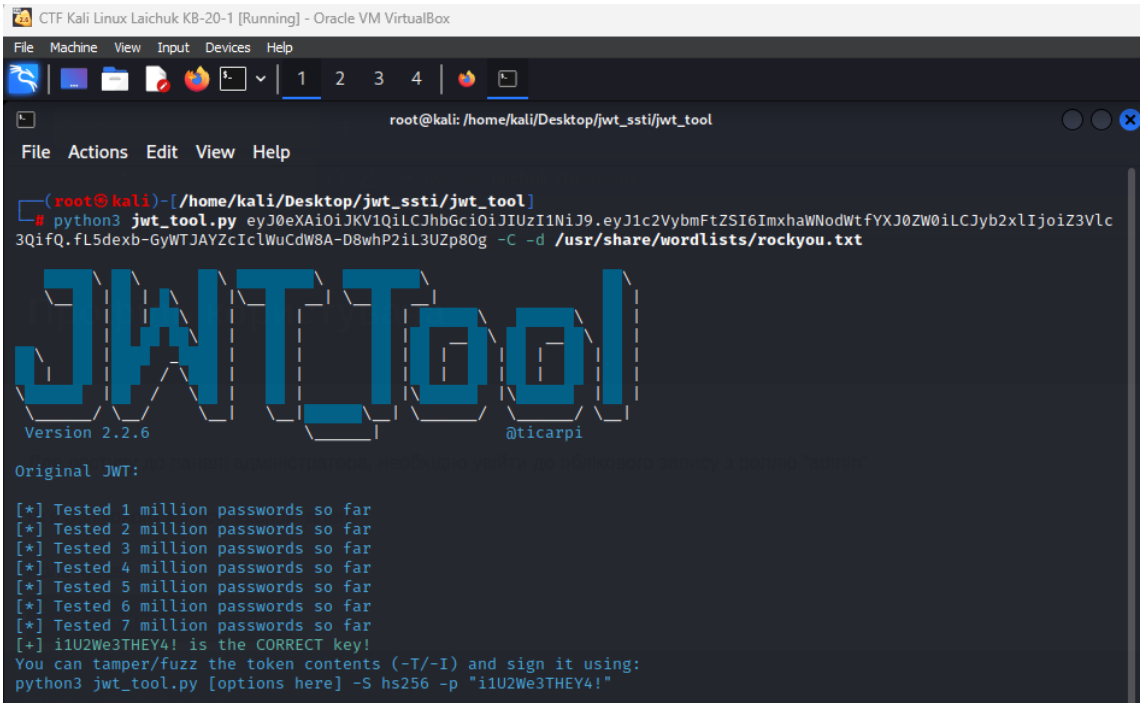


Рисунок 3.3.13 – Успішно отриманий секретний ключ для підписання токена JWT внаслідок виконання атаки за словником

В даному випадку використовувався словник “rockyou.txt”. Параметри -C та -d вказують, що необхідно провести атаку на секретний ключ, використовуючи зазначений словник.

Після отримання секретного ключа, змінимо в тілі токена значення поля “role” з “guest” на “admin” (рис. 3.3.13). Для цього використаємо веб-сайт https://jwt.io/, куди вставимо токен JWT, секретний ключ та змінимо значення поля “role”.

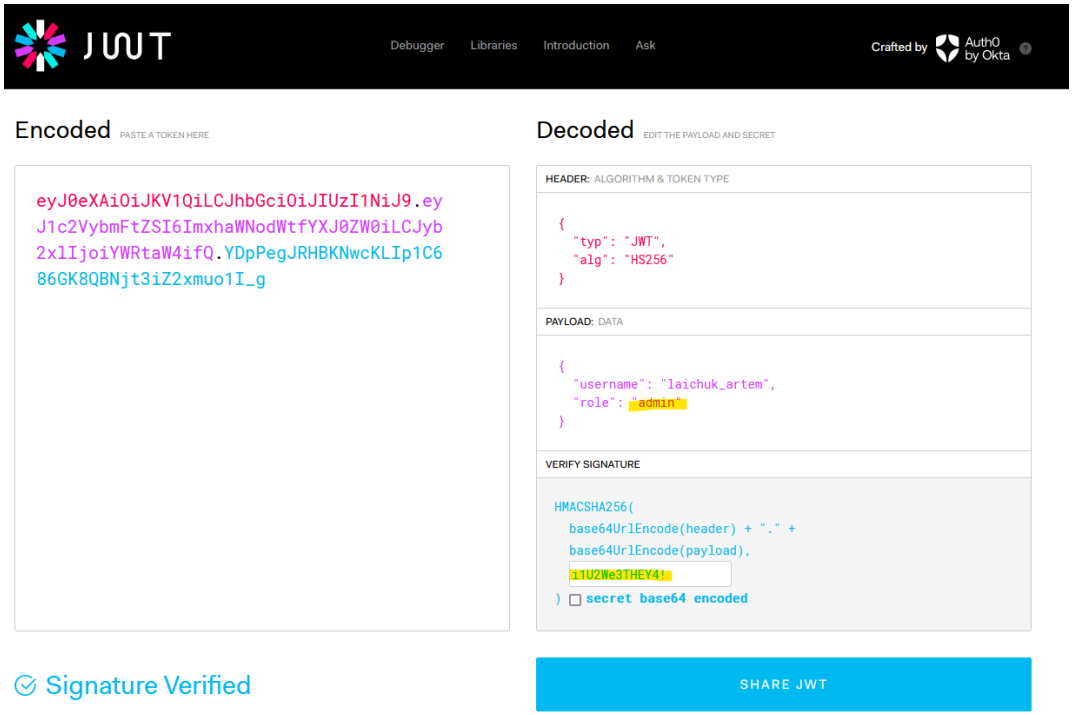


Рисунок 3.3.14 – Зміна ролі облікового запису в тілі токена JWT та його підписання за допомогою отриманого секретного ключа за допомогою атаки за словником

Далі необхідно скопіювати отриманий токен та вставити його в cookie на веб-сайті із завданням, після чого оновити сторінку, перебуваючи на сторінці профілю користувача (рис. 3.3.15).

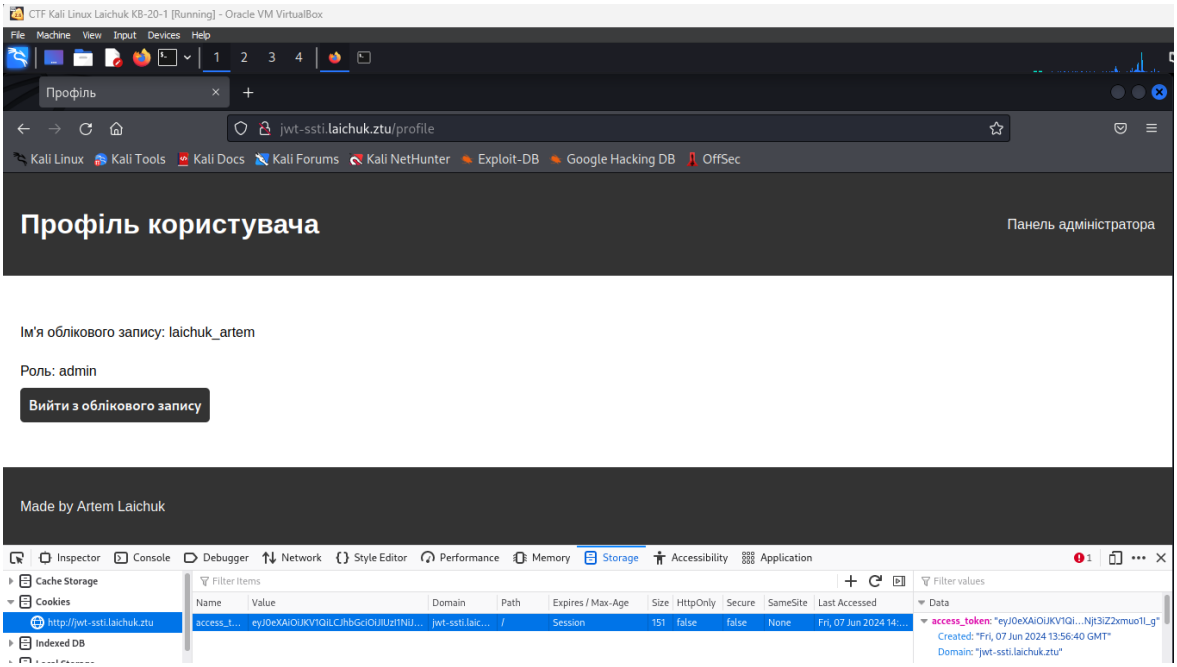


Рисунок 3.3.15 – Успішно змінена роль облікового запису з “guest” на “admin”

Як бачимо на рис. 3.3.15, після зміни токена на веб-сайті також змінилася роль облікового запису (“admin”), а також, в хедері з’явилось посилання на панель адміністратора. Перейдемо за цим посиланням.

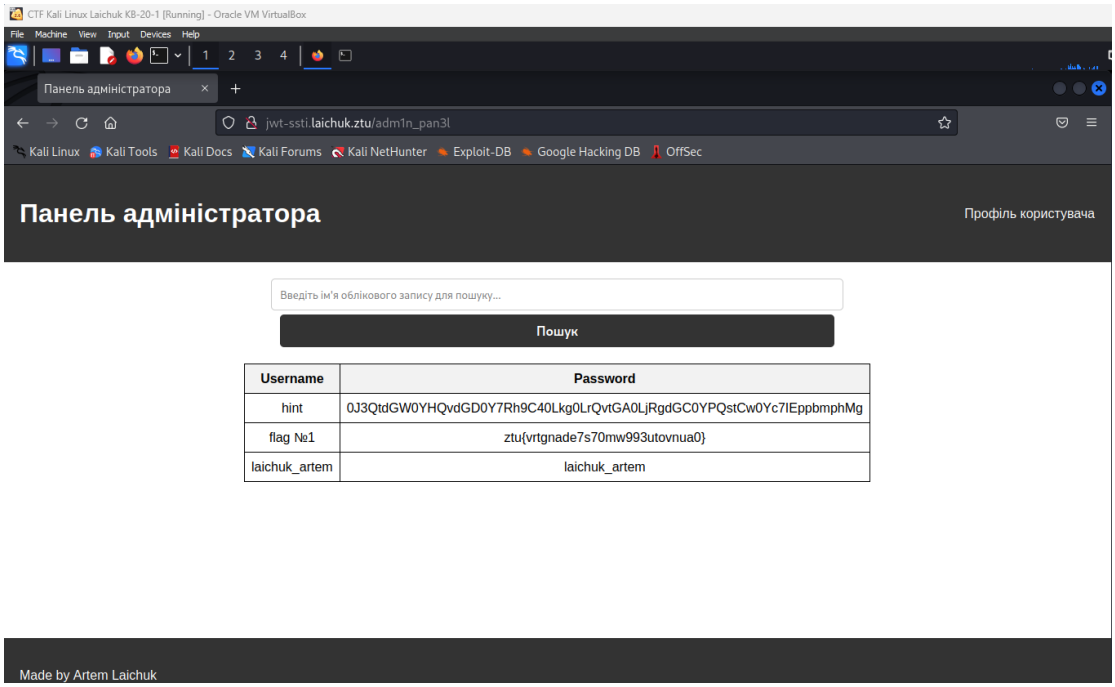


Рисунок 3.3.16 – Панель адміністратора

На панелі адміністратора, що зображена на рис. 3.3.16, можемо бачити таблицю, в якій знаходиться перший прапор, що свідчить про успішне виконання першої частини завдання. Також в таблиці є підказка, закодована за допомогою Base64. Розкодуємо дану підказку.

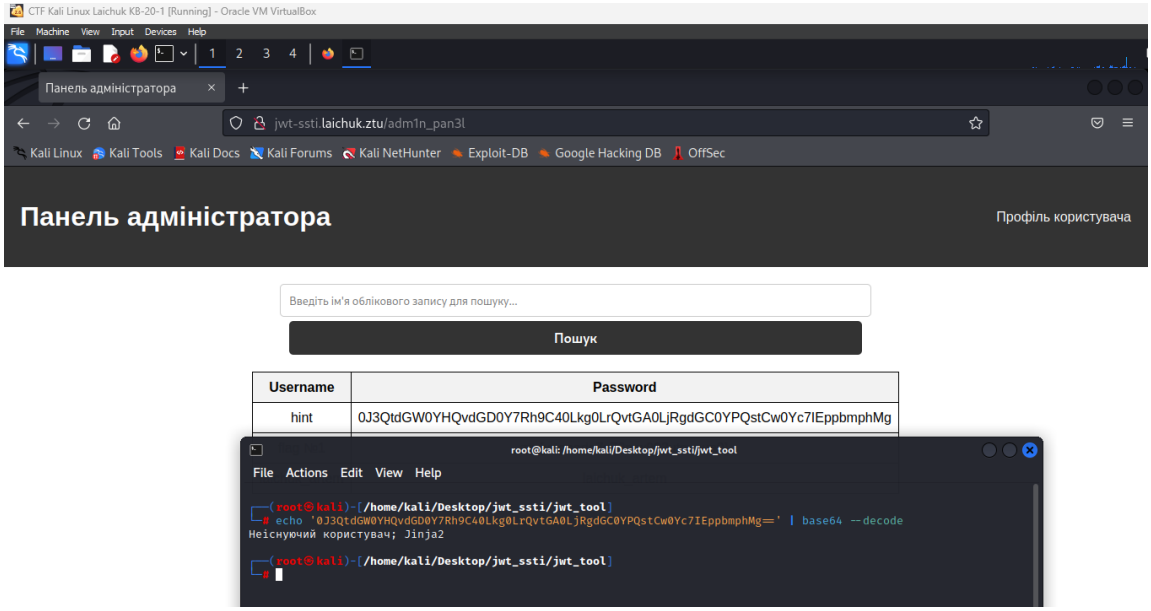


Рисунок 3.3.17 – Успішне декодування підказки

На рис. 3.3.17 зображено успішне декодування підказки, яку ми залишали на етапі розробки. Скористаємось даною підказкою та в поле пошуку користувача за вм’ям облікового запису введемо неуснуючого користувача.

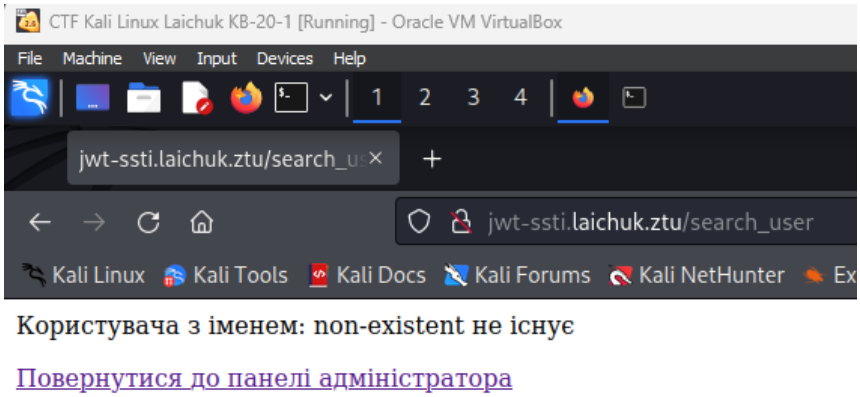


Рисунок 3.3.18 – Спроба пошуку користувача за неуснуючим ім’ям

Внаслідок пошуку користувача за неуснуючим ім’ям, з’являється сторінка, на якій зазначено те, що такого користувача не існує. Використавши другу підказку (“Jinja2”) та будь-яку пошукову систему, одразу відображається інформація про вразливості шаблонізатора Jinja2, а саме SSTI (Server-Side Template Injection). Для того, щоб перевірити чи є поле пошуку вразливим, замість імені користувача введемо “{{7*7}}” [17]. Якщо повідомлення відобразить значення “49”, то дане поле пошуку є вразливим (рис. 3.3.19 - 3.3.20).

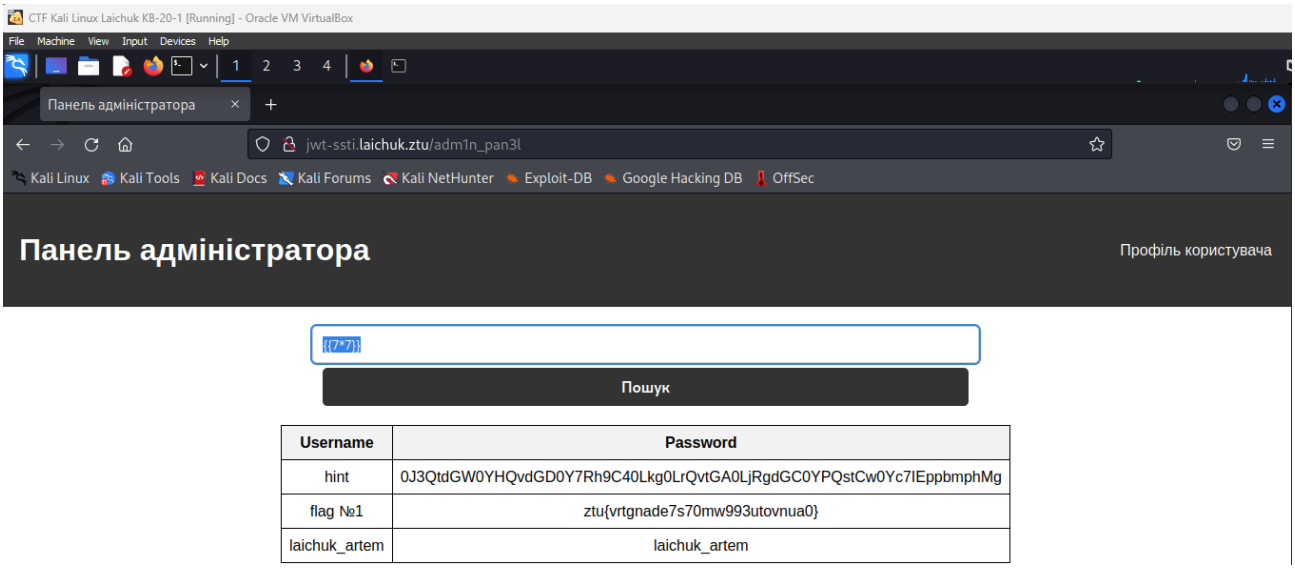


Рисунок 3.3.19 – Введення виразу {{7*7}} в шаблон для пошуку за ім’ям користувача

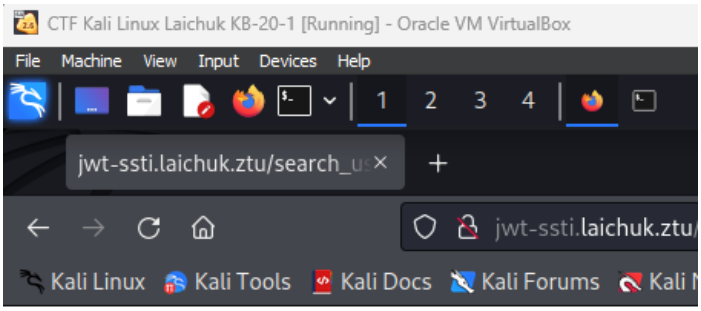


Рисунок 3.3.20 – Результат введення виразу {{7*7}} в шаблон для пошуку за ім’ям користувача

Результат, що зображений на рис. 3.3.20, свідчить про те, що поле пошуку є вразливим до атаки SSTI. Для отримання прапора, учаснику необхідно скласти шаблонний вираз, який буде виконувати довільний Python-код на сервері. Якщо учасник не має достатніх навичок програмування, базовий шаблон для такої ін’єкції можна знайти за допомогою пошуку в Інтернеті (наприклад, на веб-сайті “HackTrikcs”). Для отримання прапору, такий вираз має містити наступні компоненти:

– “request.application.__globals__” - це видає глобальний контекст виконання програми, який містить всі глобальні змінні і функції, доступні у поточній програмі.

– __buitlins__ - це вбудований модуль у Python, який містить всі вбудовані функції і об'єкти Python, такі як ‘print’, ‘len’, ‘type’ тощо.

– __import__(‘os’) - це імпортує модуль ‘os’, який надає функціонал для взаємодії з операційною системою.

– os.popen(‘команда_для_виконання_на_сервері’) – функція з модуля ‘os’, яка створює новий підпроцес, виконує команду, передану у рядку, і повертає об'єкт-потік, який можна використовувати для читання виводу команди.

– read() – читання вихідних даних з результуючого потоку.

Використовуючи вищеописані модулі та функції, отримуємо наступний вираз для виконання довільних команд на сервері: request.application.__globals__.__builtins__.__import__('os').popen(команда').read()

Виконаємо пошук файлів, що містять в своїй назві слово “flag”: {{request.application.__globals__.__builtins__.__import__('os').popen('find . -type f -iname "flag*"').read()}}. Виконаємо пошук користувача, де в ім’я передамо даний вираз (рис. 3.3.21 – 3.3.22).

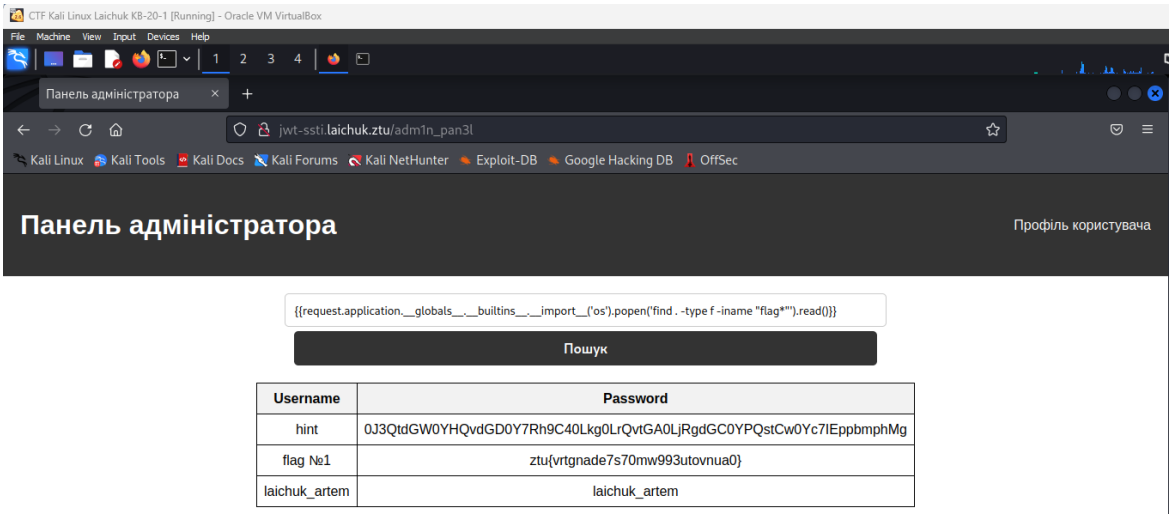


Рисунок 3.3.21 – Вставка виразу в шаблон, що виконає команду “find” на сервері для знаходженя файлів, які містять в назві слово “flag”

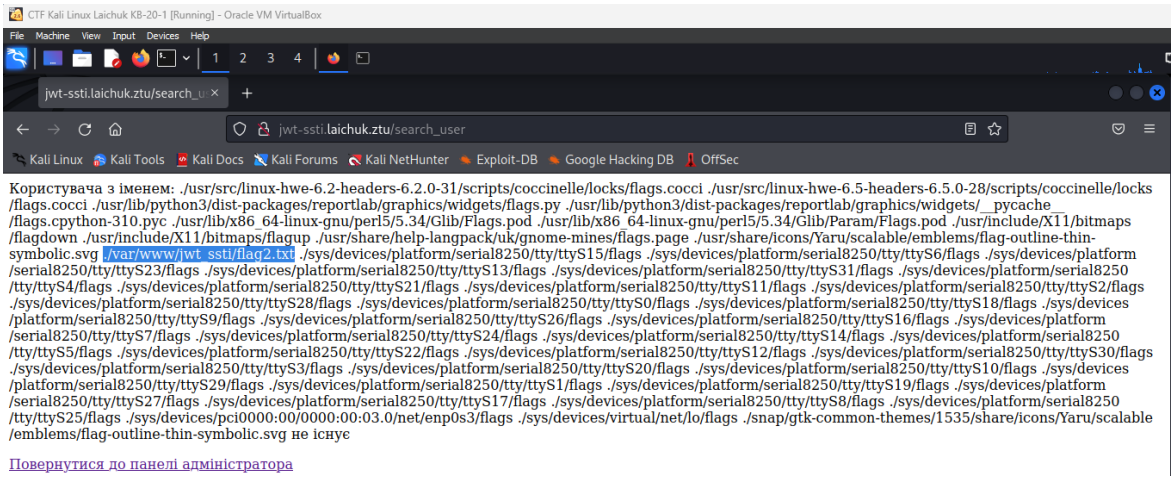


Рисунок 3.2.22 – Успішне виконання атаки SSTI

На рис. 3.2.22 бачимо, що серед усього виводу команди “find” є файл з назвою “flag2.txt”, що знайходиться за шляхом “ ./var/www/jwt_ssti/”. Спробуємо вивести вміст цього файлу на екран за допомогою команди “cat” (рис. 3.2.23 – 3.2.24).

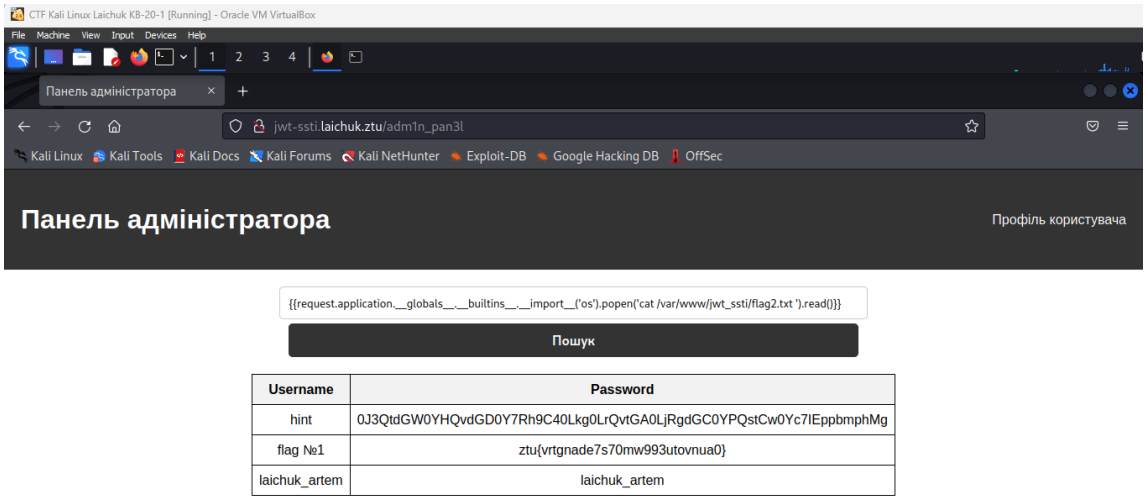


Рисунок 3.2.23 – Вставка виразу в шаблон, що виконає команду “cat” на сервері для виведення вмісту файлу “flag2.txt”

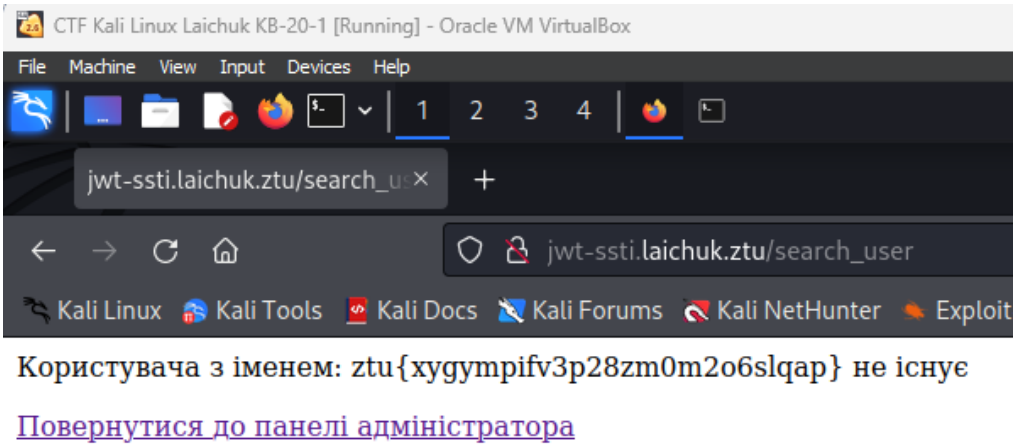


Рисунок 3.2.24 – Успішно отриманий прапор

На рис. 3.2.24 бачимо другий прапор, отриманий внаслідок виконання атаки SSTI. В результаті експлуатації вразливостей було отримано два прапора, що свідчить про успішне виконання даного завдання.

3.4. Розробка веб-додатку з вразливістю CVE-2021-22204 та її експлуатація.

3.4.1. Створення веб-сайту.

Для початку необхідно створити каталоги та файли для подальшої розробки веб-сайту. Файл, в якому буде написана логіка для веб-сайту, матиме назву “upload.php”. Також створимо каталог /uploads, який буде зберігатися зображення, що завантажуватимуть користувачі.

Загалом, файлова структура виглядає наступним чином:

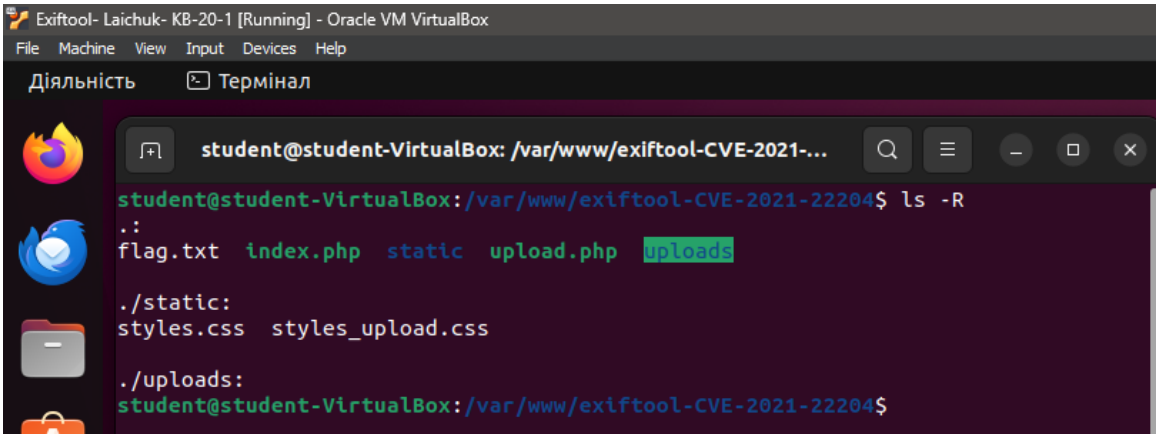


Рисунок 3.4.1 – Файлова структура проекту

Для початку необхідно розробити функціонал завантаження зображень з вебсайту на сервер. Також необхідно реалізувати фільтрацію назви зображення що завантажується для запобігання атаки Command Injection, а також перевірку типу файлу, дозволивши лише розширення .jpeg, .jpg та .png.

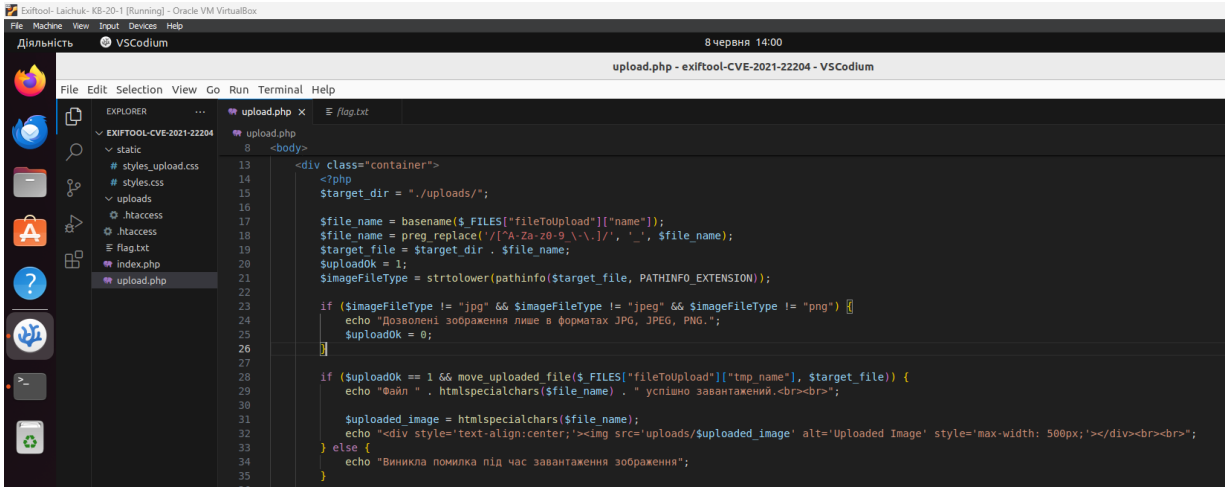


Рисунок 3.4.2 – Фрагмент коду, що відповідає за завантаження зображень на сервер

На рис. 3.4.2 зображена фільтрація назви завантажуваного файлу за допомогою регулярного виразу, який замінює всі символи, окрім латинських букв, цифр, підкреслення, дефіса і крапки, на підкреслення “_”. Далі перевіряється, чи розширення файлу відповідає одному з дозволених форматів (jpg, jpeg, png). Якщо ні, виводиться повідомлення про помилку. Якщо зображення пройшло усі перевірки, то виводиться повідомлення про успішне завантажування файлу, а також використовується “htmlspecialchars()”, щоб запобігти XSS-атакам при відображенні імені файлу.

За виконання команди “exiftool” та виведення EXIF-даних на веб-сайті відповідає наступний фрагмент коду (рис. 3.4.3).

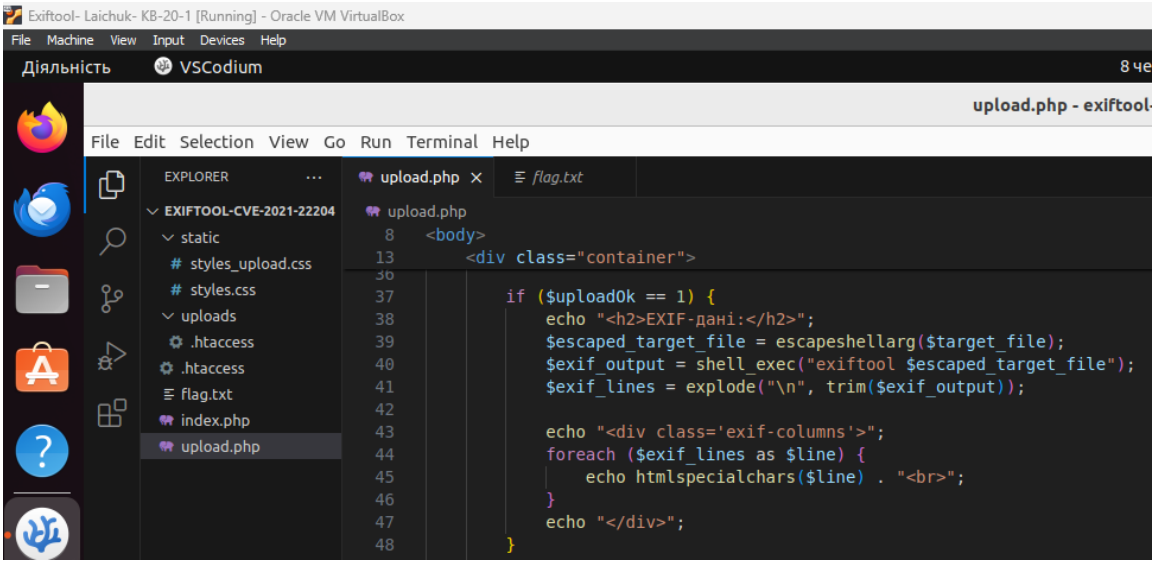


Рисунок 3.4.3 – Фрагмент коду, що відповідає за виконання команди “exiftool” та виведення результату виконання на веб-сайт

На рис. 3.4.3 зображено, що ім’я файлу екранується для безпечного використання в командному рядку за допомогою “escapeshellarg()”, що запобігає можливим ін’єкціям командного рядка. Для виконання зовнішньої команди “exiftool” використовується “shell_exec()”, після чого дані відображаються на сторінці у вигляді окремих рядків, кожен з яких проходить через функцію “htmlspecialchars()”, щоб уникнути проблем із безпекою.

Вразливістю є саме версія встановленого інструмента “Exiftool” – 12.23.

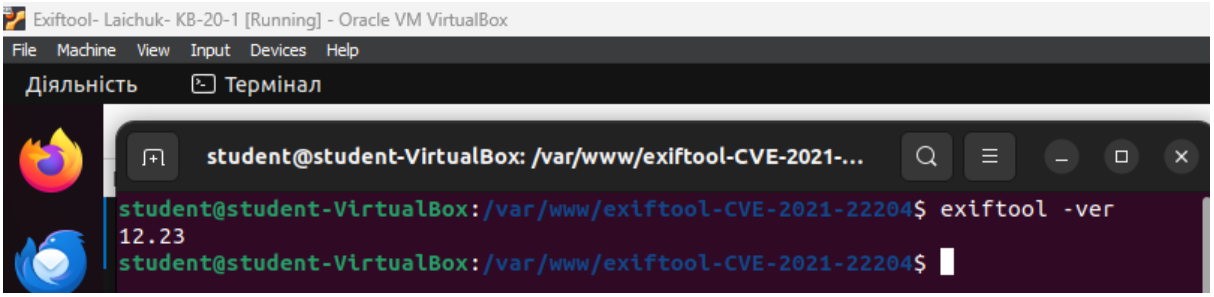


Рисунок 3.4.4 – Вразлива версія встановленого інструмента Exiftool

Після виконання аналогічних дій щодо створення віртаульного хоста на веб-сервері Apache, описаних при створенні першого завдання (рис. 3.2.8), перейдемо до тестування створеного веб-сайту та експлуатації вразливості.

3.4.2. Тестування розробленого функціоналу та експлуатація реалізованої вразливості.

З робочої станції Kali Linux спробуємо відкрити веб-сайт за допомогою браузера.

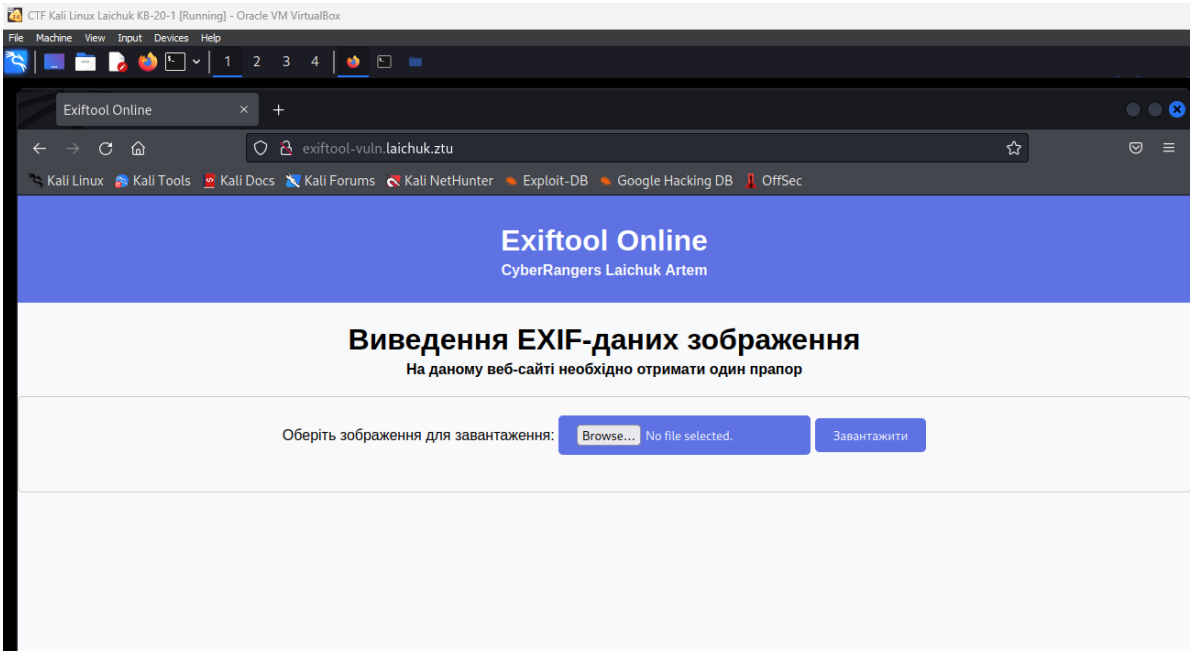


Рисунок 3.4.5 – Головна сторінка веб-сайту

Для перевірки функціоналу завантажимо будь-яке зображення та перевіримо коректність виведення його EXIF-даних.

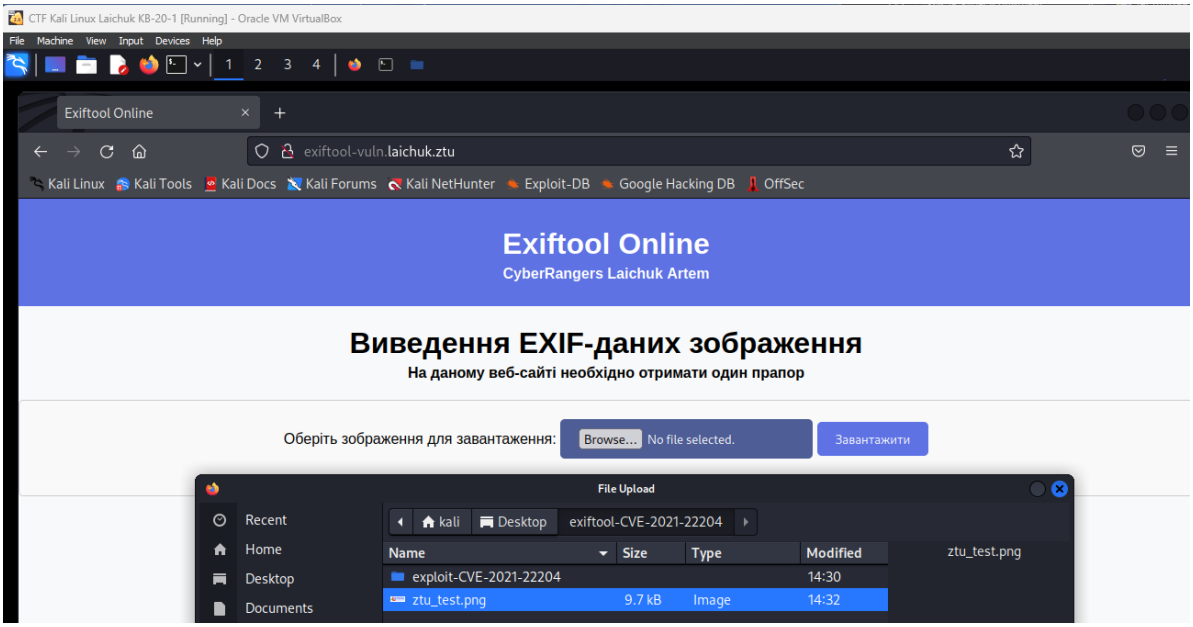


Рисунок 3.4.6 – Вибір зображення для завантаження на веб-сервер

Після вибору зображення необхідно натиснути на кнопку “Завантажити”.

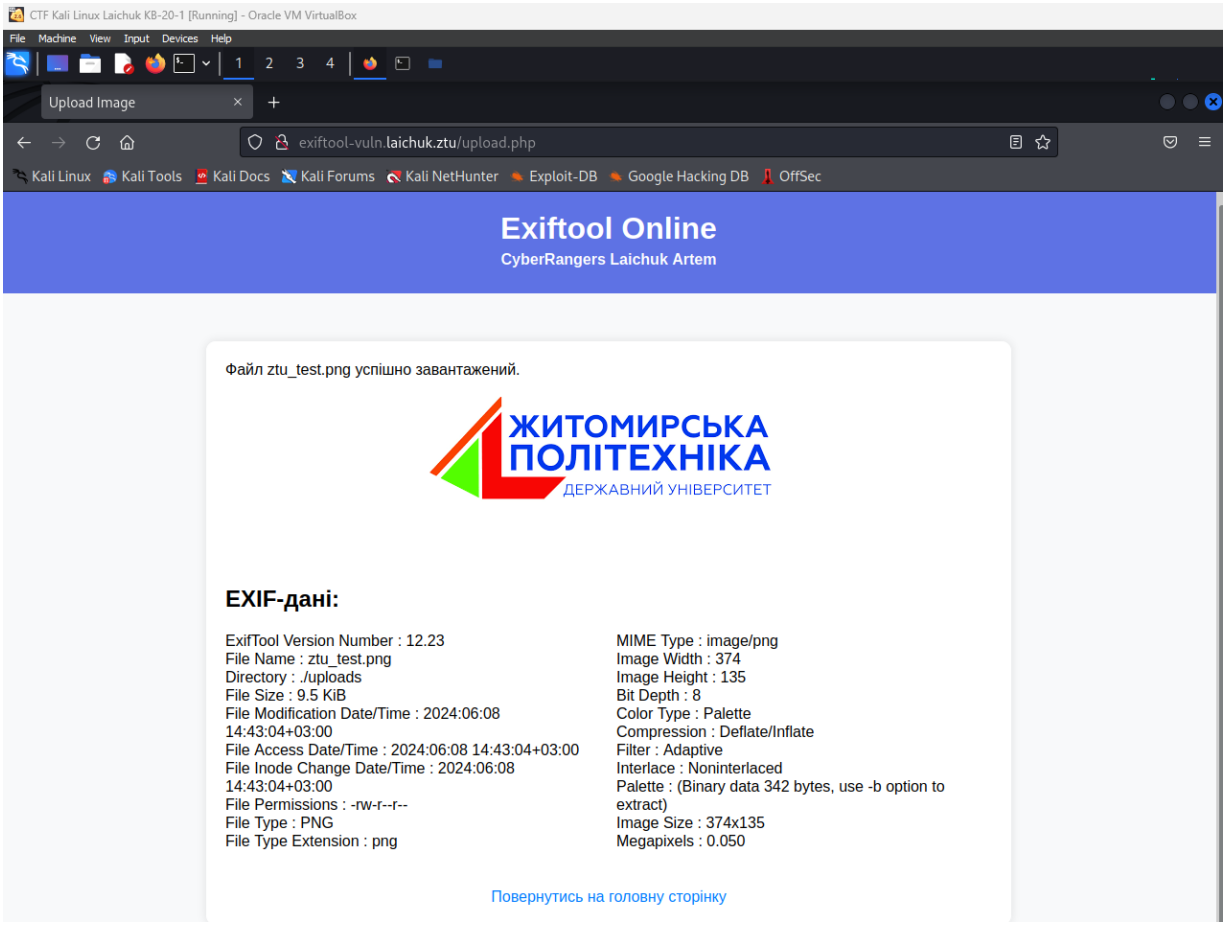


Рисунок 3.4.7 – Успішно завантажене зображення на веб-сервер та виведення його EXIF-даних

На рис. 3.4.7. бачимо, що зображення успішно завантажилось на веб-сервер та EXIF-дані відобразились на сторінці. Це свідчить про те, що веб-сайт працює коректно. Також, в EXIF-даних бачимо, що використовується версія ExifTool 12.23.

Наступним кроком є проведення атаки. Для цього використаємо інструмент exploit-CVE-2021-22204 (https://github.com/UNICORDev/exploit-CVE-2021-22204).

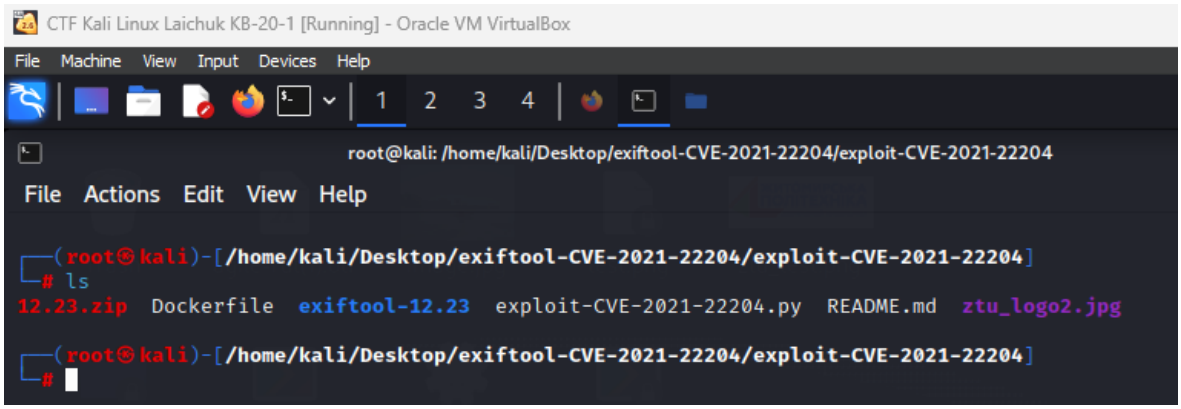


Рисунок 3.4.8 – Вміст каталогу, необхідного для проведення атаки

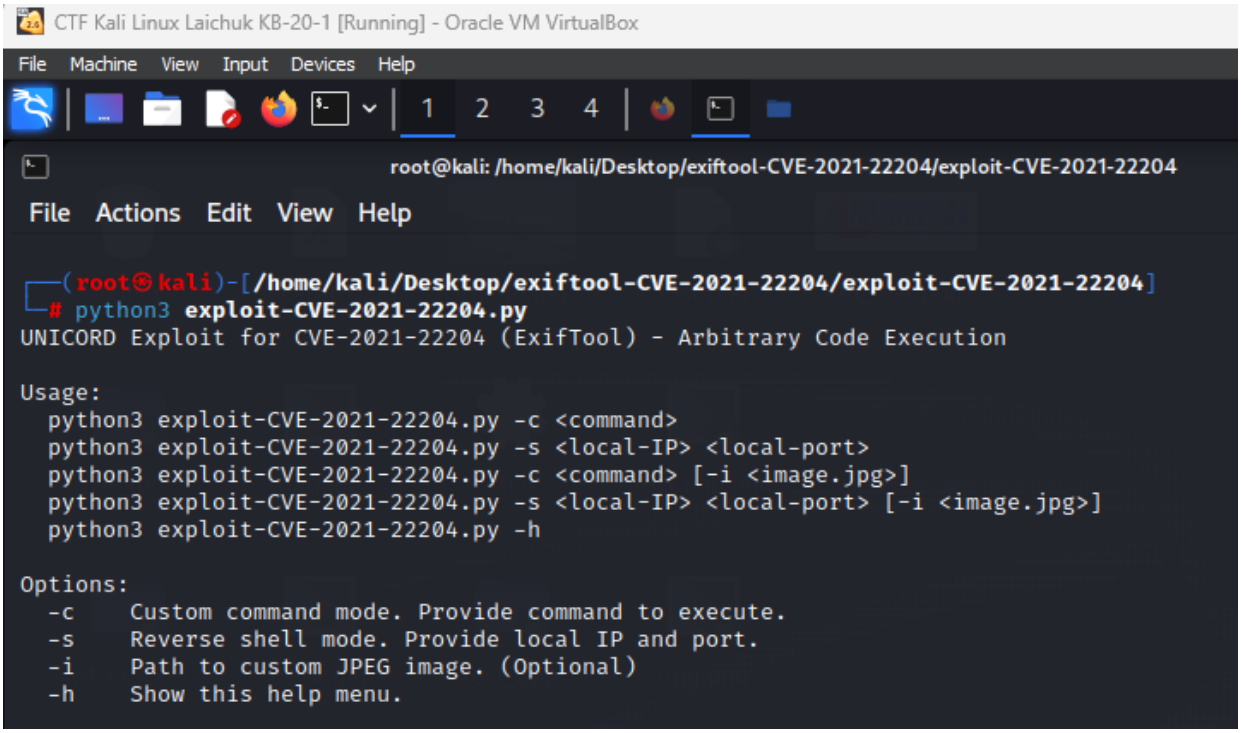


Рисунок 3.4.9 – Перелік можливих параметрів для виконання атаки

В даній роботі виконаємо два варіанти атаки з використанням Command Injection та зворотньої оболонки. Для виконання атаки, пов'язаної з впровадженням команд в зображення, скористаємось параметром -c та -i для визначення зображення, в яке необхідно впровадити команду.

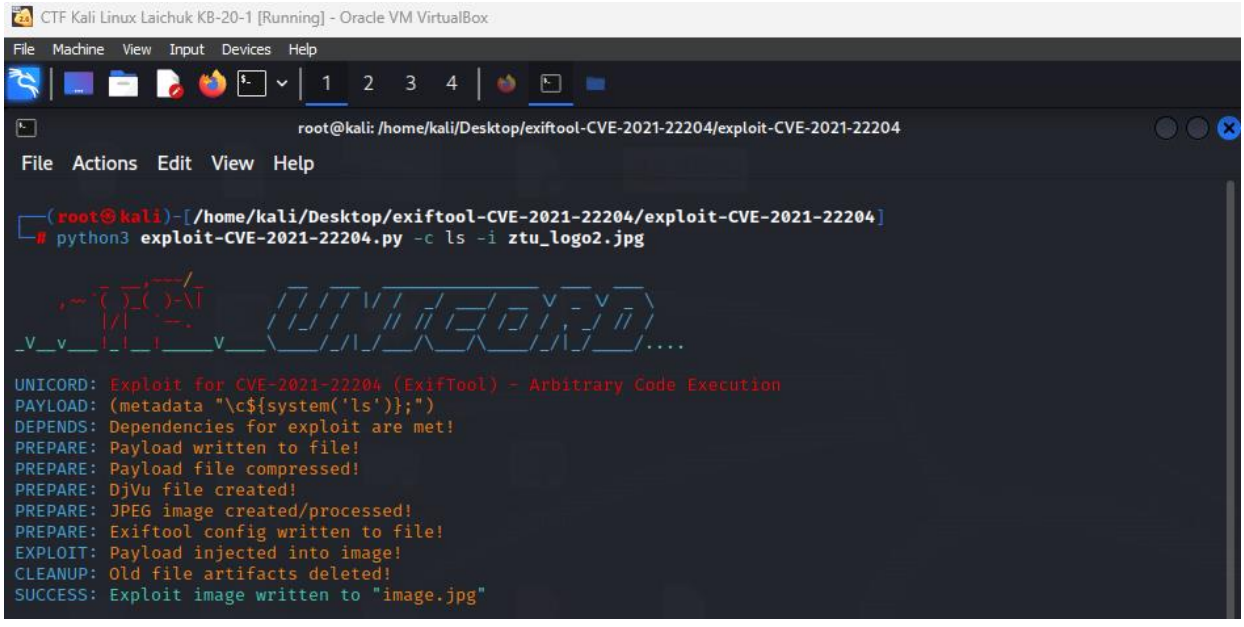


Рисунок 3.4.10 – Успішне впровадження команди "ls" в зображення

Після виконання скрипту для впровадження команди в зображення, на виході отримуємо шкідливе зображення "image.jpg", яке необхідно завантажити на веб-

сервер та отримати відповідь з переліком файлів та каталогів, що знаходяться в поточному каталозі (впроваджена команда “ls”).

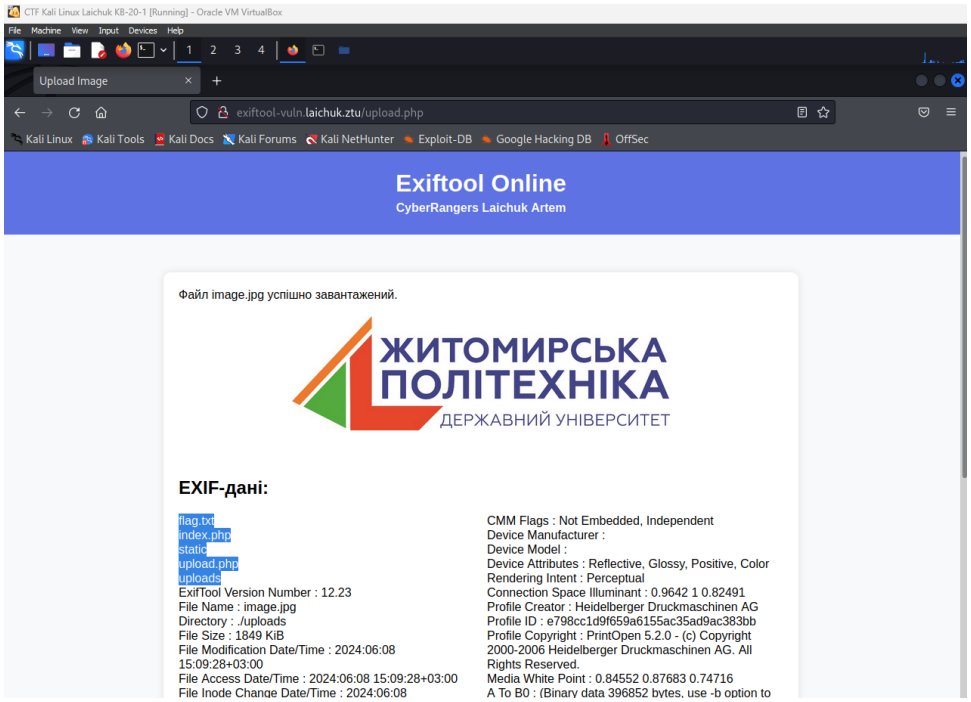


Рисунок 3.4.11 – Успішне завантаження шкідливого зображення та отримання переліку файлів та каталогів на веб-сервері

Прочитаємо файл “flag.txt”, впровадивши в зображення команду “cat flag.txt”.

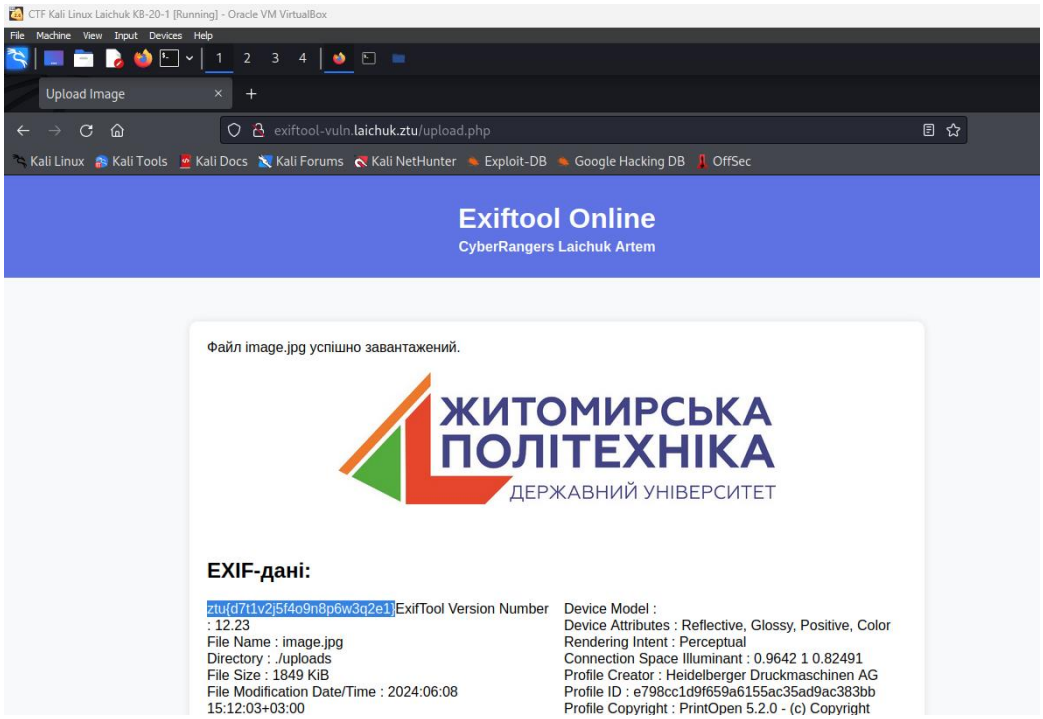


Рисунок 3.4.12 – Успішно захоплений прапор

Як бачимо на рис 3.4.12, прапор був успішно захоплений. Також виконаємо атаку з використанням зворотньої оболонки. Для цього використаємо параметр -s, якому передамо IP-адресу та порт робочої станції учасника.

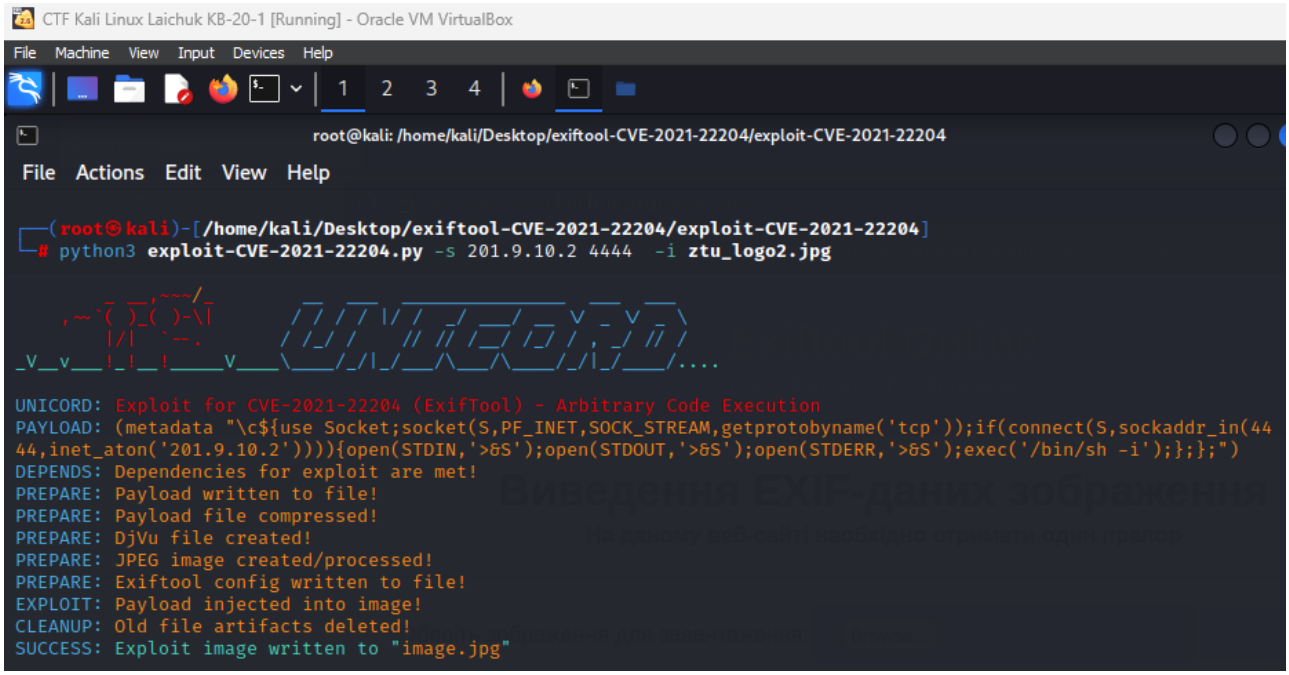


Рисунок 3.4.13 – Впровадження команди для отримання зворотньої оболонки від веб-сервера до робочої станції Kali Linux

Наступним кроком є встановлення IP-адреси та порта на прослуховування для подальшого отримання підключення. Виконаємо це за допомогою “netcat”, виконавши команду “nc -lvp 4444” (параметри -l – listen, -v – verbose, -p - port).

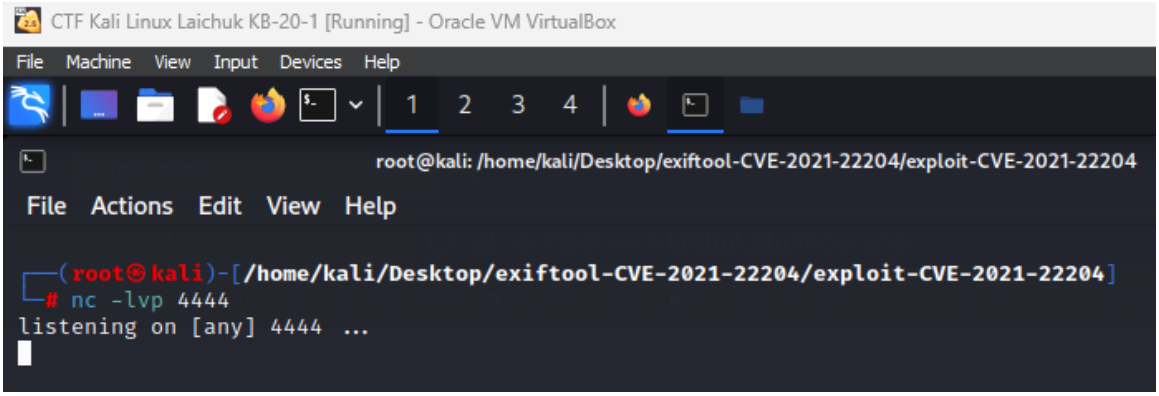


Рисунок 3.4.14 – Успішне встановлення порта 4444 на прослуховування

Фінальним кроком для отримання зворотньої оболонки є завантаження шкідливого зображення на веб-сервер.

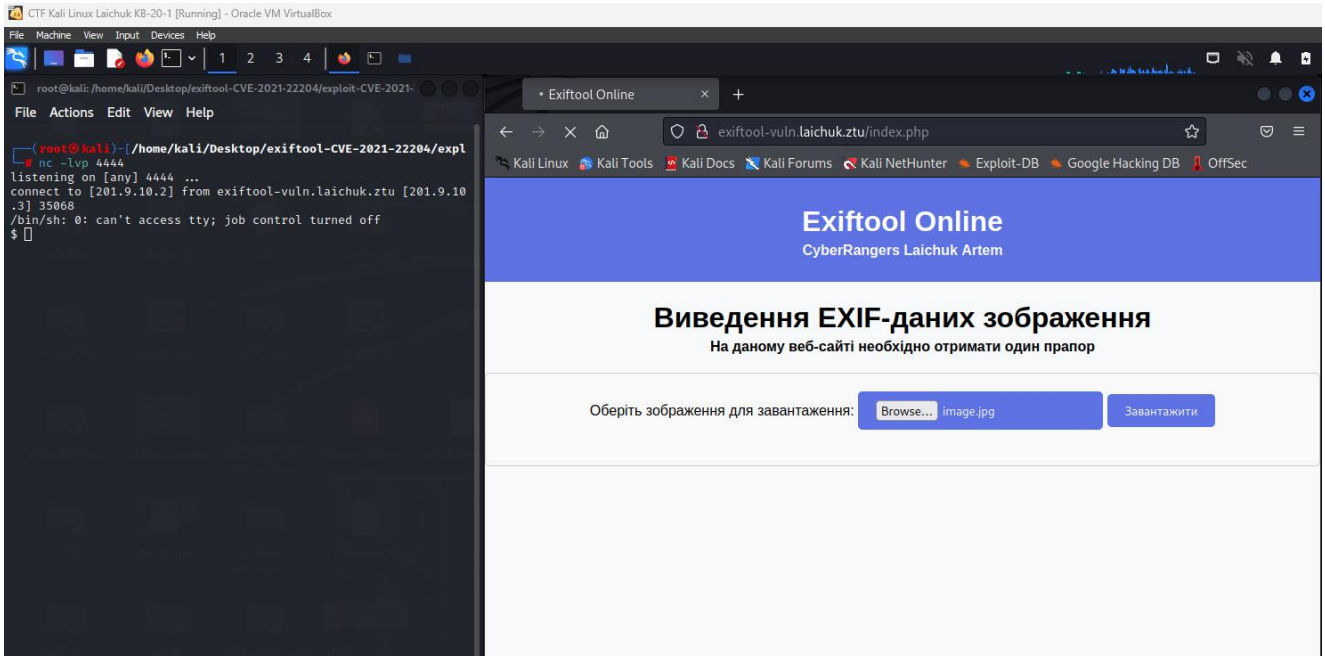


Рисунок 3.4.15 – Успішно отримана зворотня оболонка внаслідок завантаження шкідливого зображення на веб-сервер

Після отримання щворотньої оболонки необхідно прочитати вміст файлу “flag.txt” для захоплення прапора.

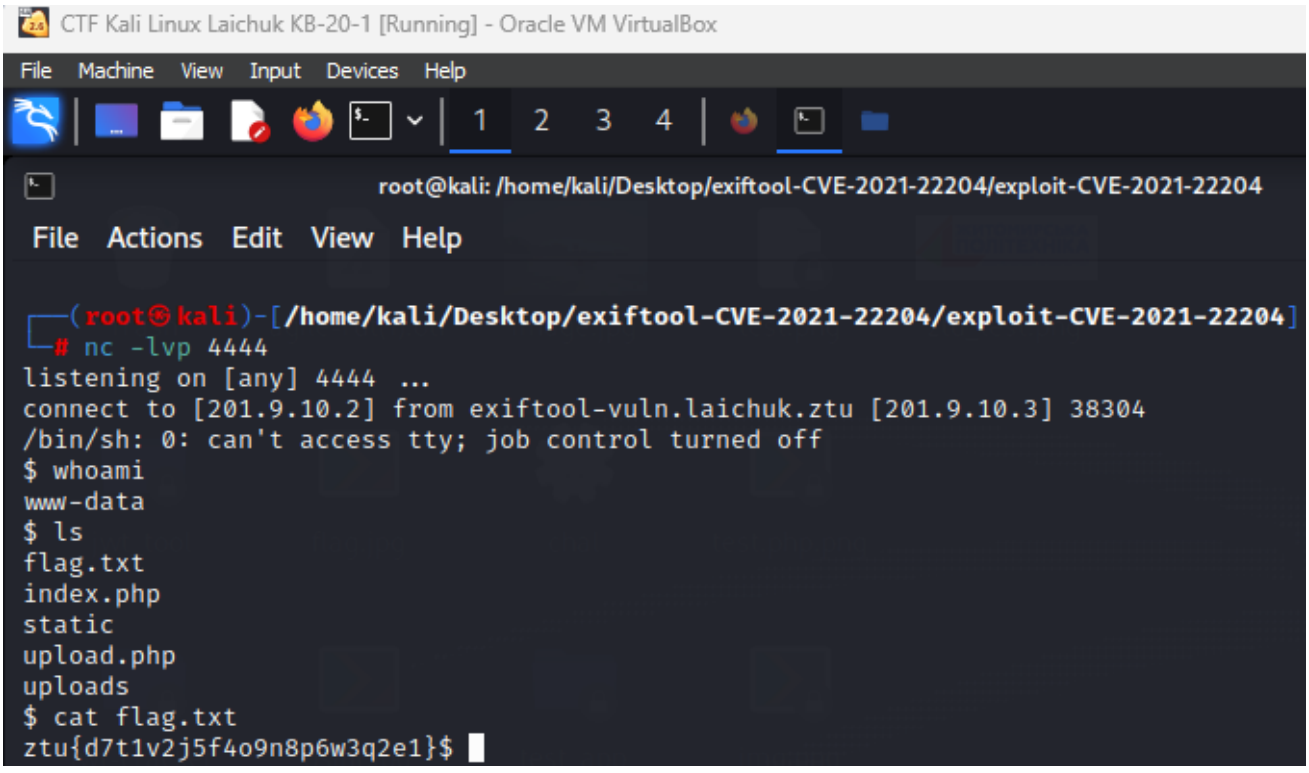


Рисунок 3.4.16 – Успішно захоплений прапор

Ще одним варіантом для виконання даного завдання є використання експлойту “exploit/unix/fileformat/exiftool_djvu_ant_perl_injection” в Metasploit. Щоб запустити Metasploit необхідно виконати команду “msfconsole”, після чого необхідно обрати потрібний нам експлойт, встановити необхідні параметри для проведення атаки (встановити DISABLEPAYLOADHANDLER у false, WfsDelay у 100 і запустити експлойт командою “exploit”) [6]. Даний модуль автоматично згенерує шкідливе зображення для отримання зворотньої оболонки. Необхідно завантажити згенероване зображення на веб-сайт та отримати підключення до веб-сервера, після чого захопити прапор (рис. 3.4.17).

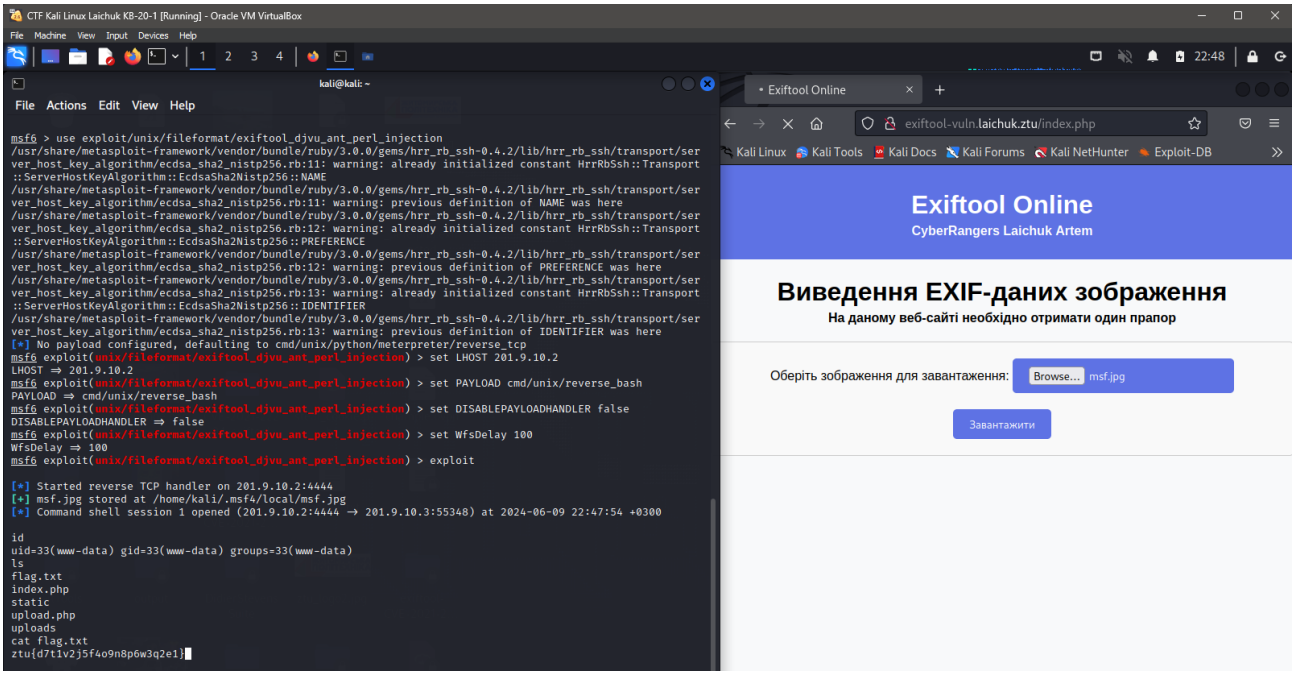


Рисунок 3.4.17 – Успішно отримана зворотня оболонка за допомогою інструмента Metasploit

Як бачимо на рис. 3.4.17, модуль в Metasploit автоматично встановив порт 4444 на прослуховування. Після завантаження шкідливого зображення на веб-сайті, на робочу станцію Kali Linux надійшло зворотнє підключення, яке дозволяє виконувати команди на сервері. Захоплення прапора свідчить про успішну експлуатації вразливості CVE-2021-22204.

3.5. Розробка веб-додатку з вразливостями SQLi та Cross-Site Scripting, тестування та експлуатація.

3.5.1. Створення веб-сайту.

Для початку необхідно під’єднатись до сервера MySQL, створити користувача, базу даних та таблиці.

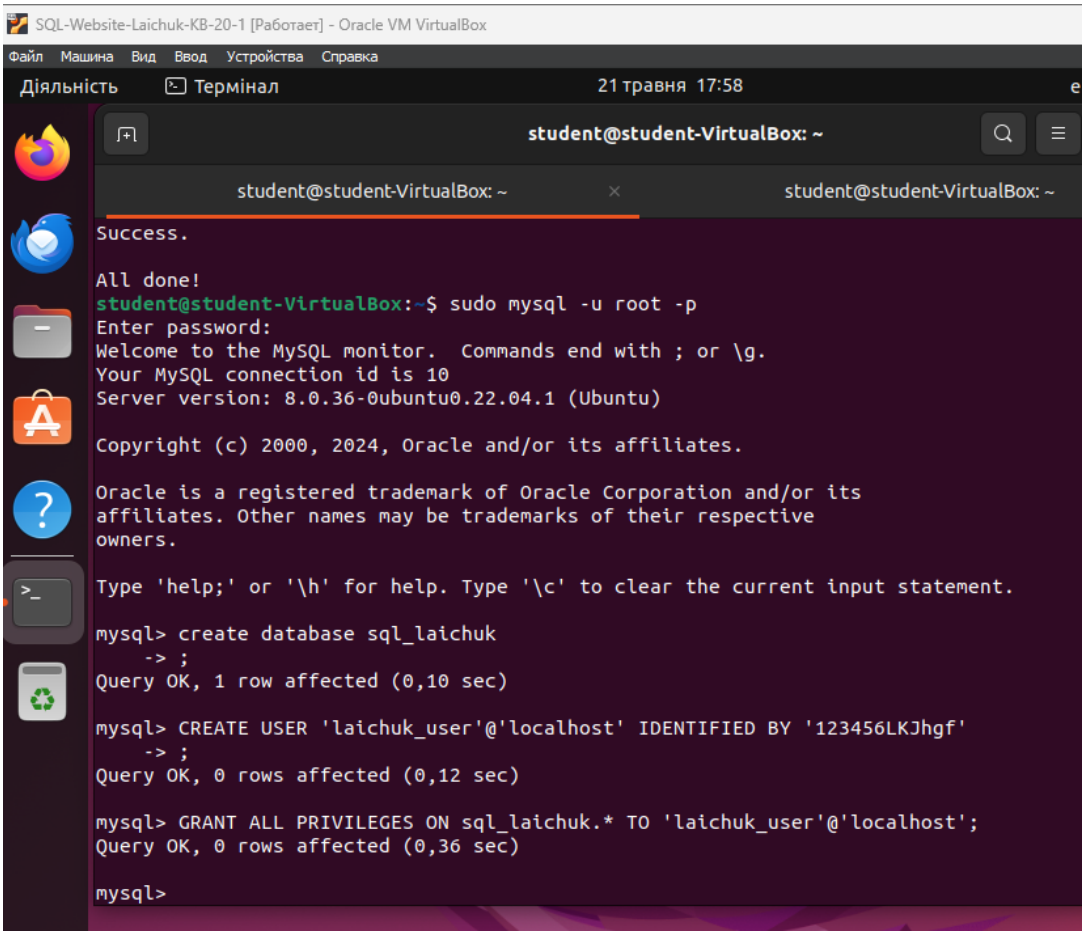


Рисунок 3.5.1 – Створення бази даних “sql_laichuk” та користувача “laichuk_user” з подальшим наданням повних прав на управління базою даних

Скрипт створення таблиць буде винесено в додатки до кваліфікаційної роботи. В результаті налаштування бази даних для проекту було створено т наповнено 4 таблиці: “users”, “posts”, “comments” та“flags”, які були описані у п. 2.3.3.

Також, перед розробкою проекту необхідно розробити його структуру. Для реєстрації та входу в обліковий запис створимо файли “register.php” та “login.php” відповідно. Для з’єднання з базою даних створимо файл “db.php”, а також “post.php”, “view_post.php” та “comments.php” для взаємодії користувачів з записами на веб-сайті.

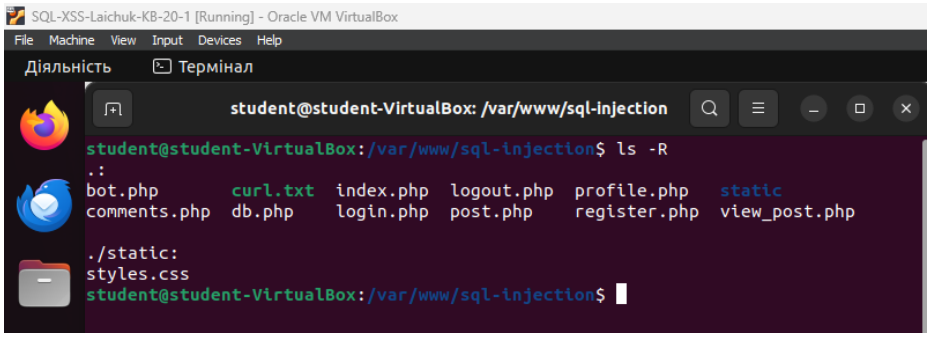


Рисунок 3.5.2 – Файлова структура проекту

Далі розглянемо найважливіші функції та вразливості в коді для подальшої їх експлуатації. Весь код проекту також буде винесено в додатки. Підключення до бази даних відбувається за допомогою файлу “db.php”.

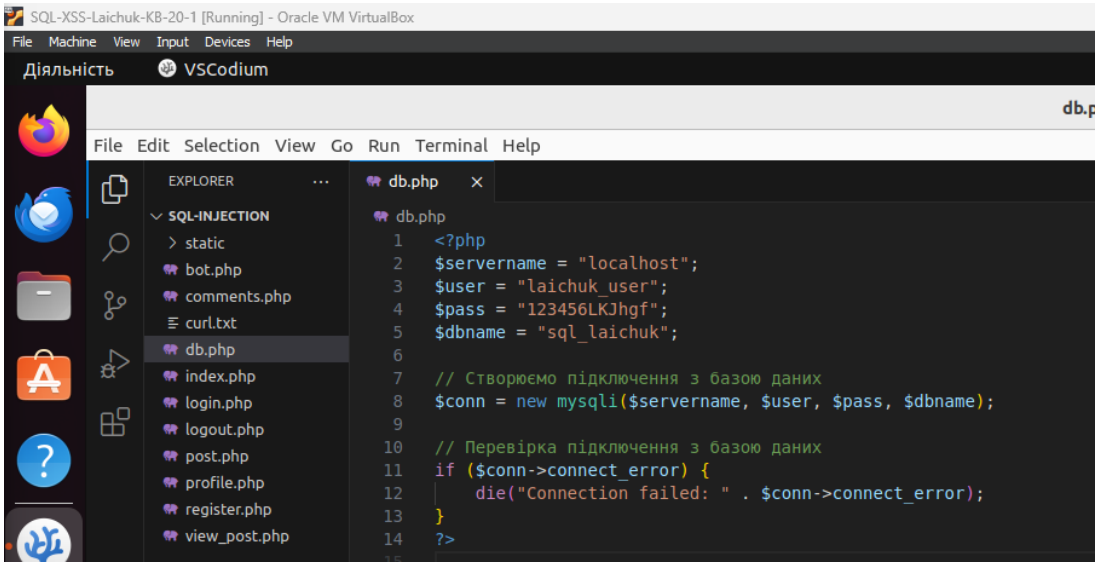


Рисунок 3.5.3 – Фрагмент коду, що відповідає за підключення до бази даних

Перша вразливість – SQLi, реалізована в файлі view_post.php.

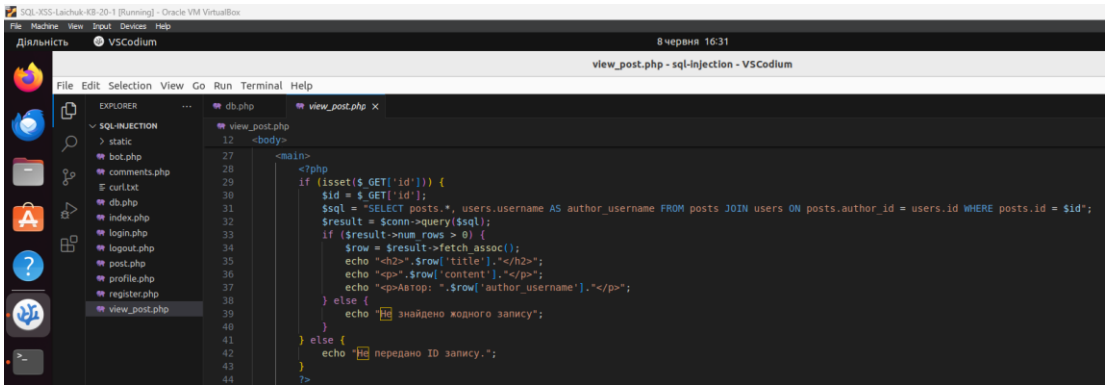


Рисунок 3.5.4 – Фрагмент коду із реалізованою вразливістю SQLi

Фрагмент коду, що зображений на рис. 3.5.4, перевіряє чи параметр “id” переданий через GET-запит, і якщо так, використовує його для виконання SQL-запиту, щоб витягнути дані з таблиці “posts” і пов'язаної таблиці “users”, з'єднуючи їх по полю “author_id”. Якщо результат запиту містить записи, він виводить заголовок, вміст і ім'я автора публікації; якщо ні, виводиться повідомлення про відсутність записів. Якщо параметр “id” не переданий, виводиться повідомлення про його відсутність. У цьому коді є вразливість до SQL-ін'єкції, оскільки значення параметра “id” вставляється безпосередньо в SQL-запит без належної обробки, що дозволяє зловмиснику виконати довільний SQL-код через маніпуляцію параметром “id”.

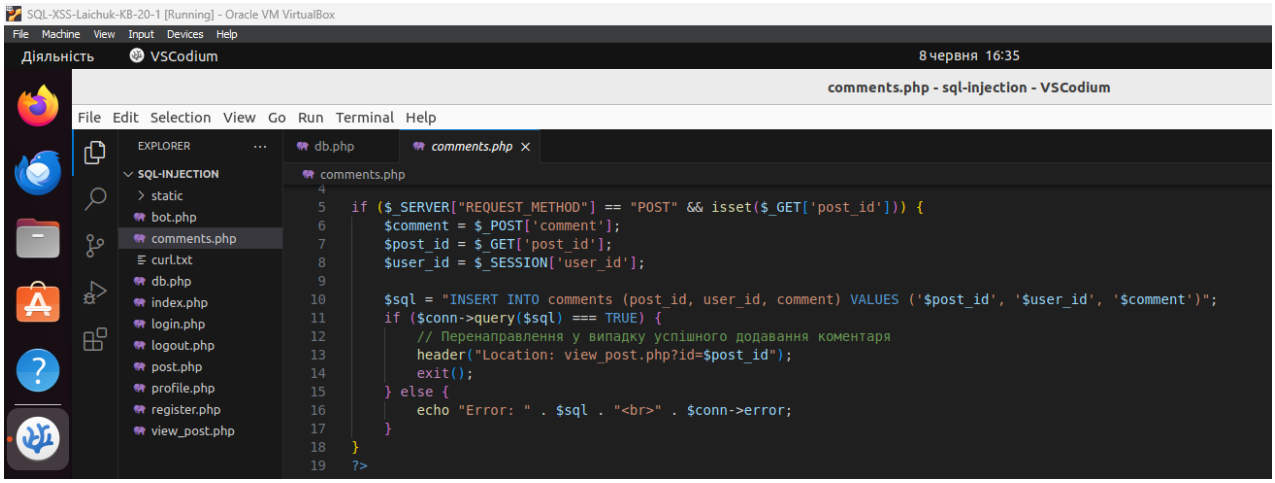


Рисунок 3.5.5 – Фрагмент коду із реалізованою вразливістю XSS

На рис. 3.5.5 зображений фрагмент коду, який відповідатиме за вразливість XSS, адже коментарі жодним чином не фільтруються, тому користувач може занести до бази даних скрипт, який виконуватиметься у кожного, хто відкриє цей коментар (Stored XSS).

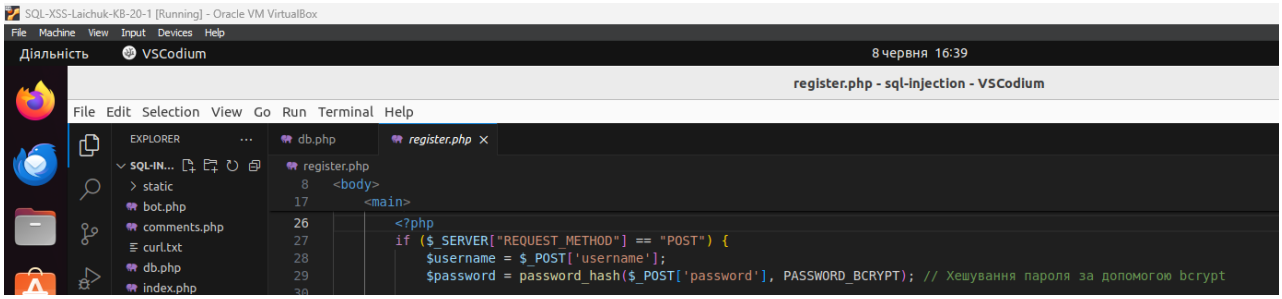


Рисунок 3.5.6 – Фрагмент коду, що відповідає за хешування паролів

На рис. 3.5.6 бачимо реалізацію хешування паролів за допомогою алгоритму bcrypt. Незважаючи на хешування, використанян слабких паролів робить облікові записи вразливими до викрадення, адже цей алгоритм піддається до атаки за словником.

Перший прапор учасник захопить отримавши доступ до бази даних, де він буде залишей в таблиці “flags”.

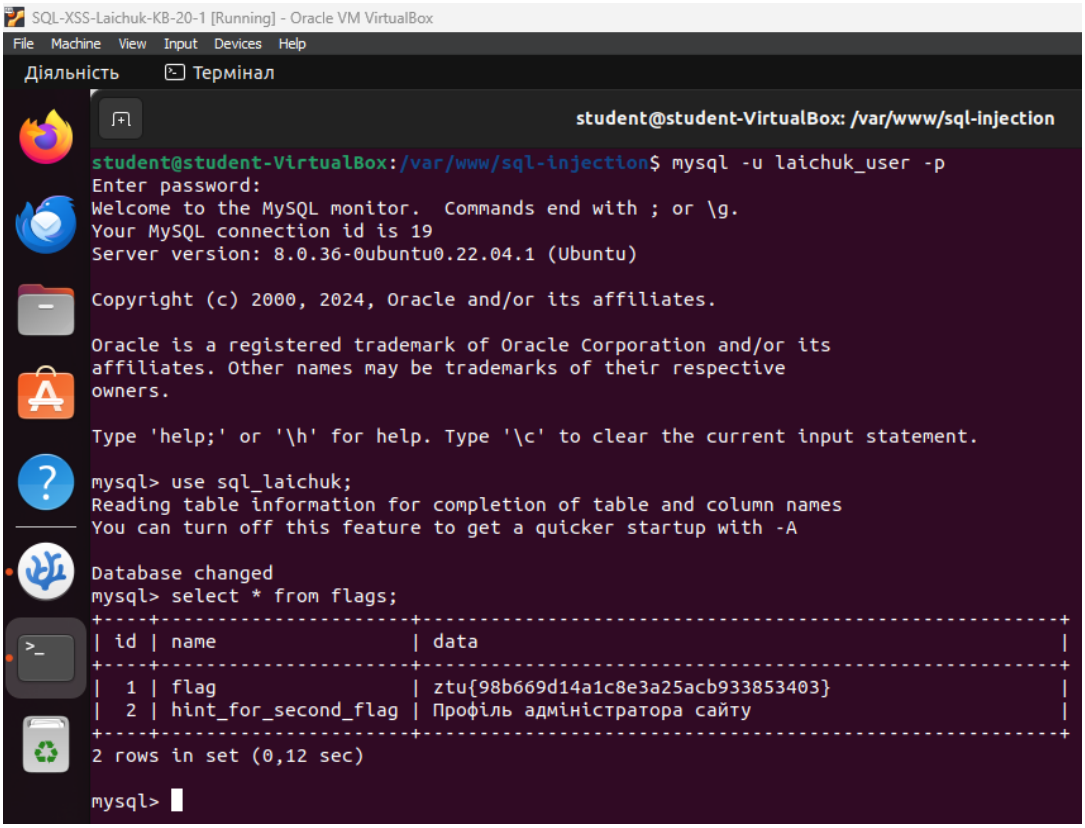


Рисунок 3.5.7 – Вміст таблиці “flags”

Другий прапор та підказку учасник отримає увійшовши на веб-сайті під обліковим записом “admin”.

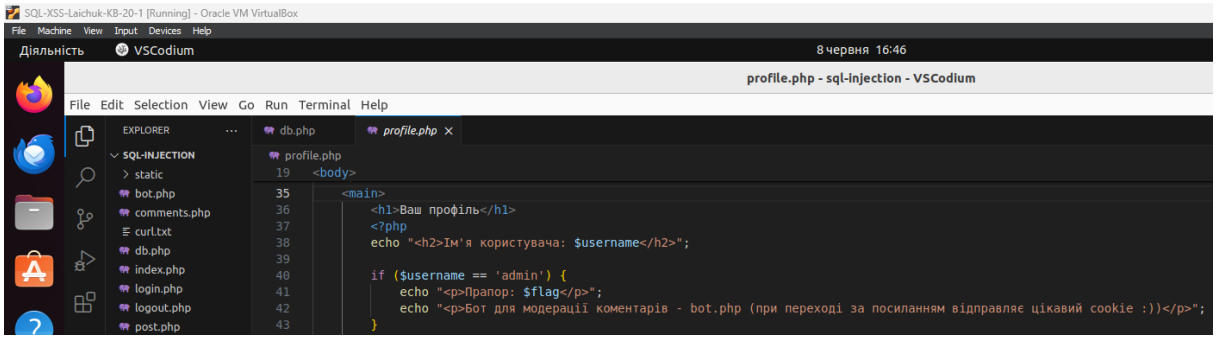


Рисунок 3.5.8 – Фрагмент коду, який відповідає за виведення другого прапору на сторінці веб-сайту

Третій прапор учасник захопить за допомогою XSS та функціоналу сторінки bot.php. При надсиланні учасником посилання на запис з шкідливим коментарем, бот проаналізує його та відправить cookie з прапором.

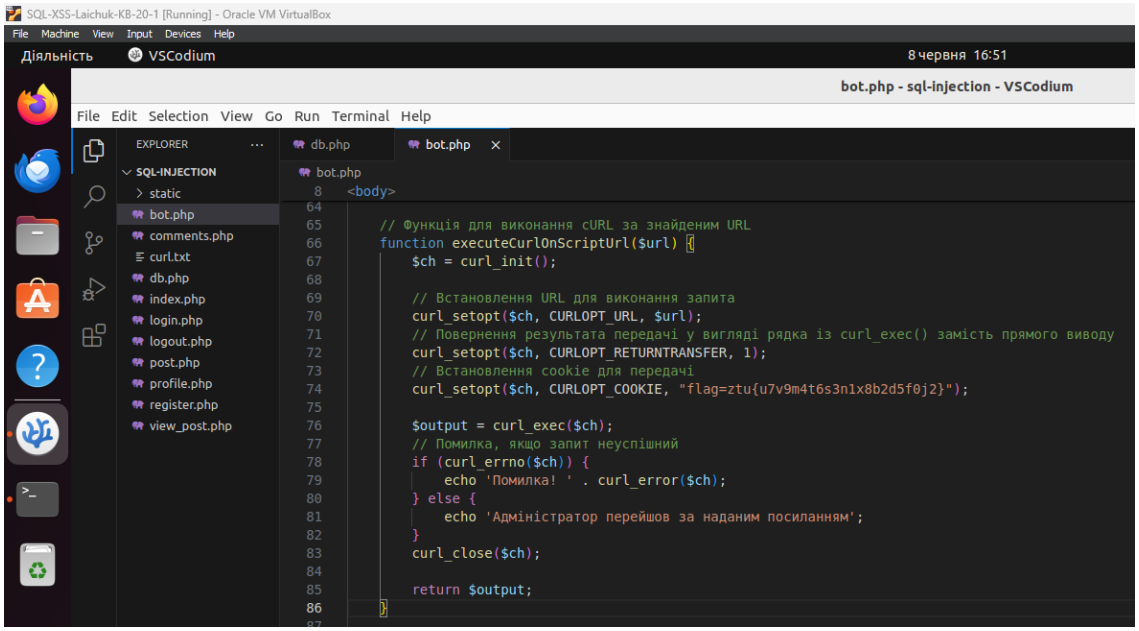


Рисунок 3.5.9 – Функція, що відповідає за відправлення cookie при переході за посиланням

Сторінка входу захищена від SQL-ін’єкцій для ускладнення завдання за допомогою використання підготовлених запитів (prepared statements) з параметризованими запитами.

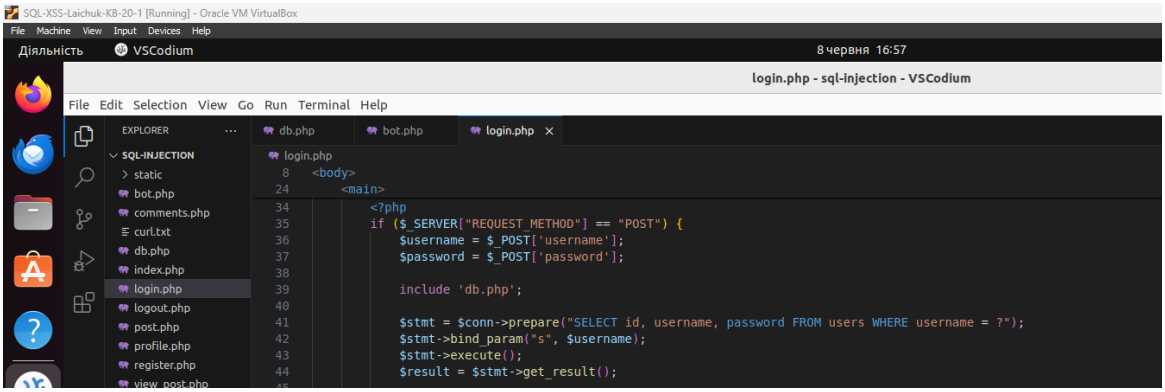


Рисунок 3.5.10 – Використання підготовлених запитів (prepared statements) з параметризованими запитами для захисту від SQL-ін’єкцій на сторінці входу в обліковий запис

Після проведення усіх вищеописаних налаштувань та написання коду перейдемо до тестування функціоналу та експлуатації вразливостей.

3.5.2. Тестування розробленого функціоналу та експлуатація реалізованих вразливостей.

В браузері на робочій станції Kali Linux відкриємо веб-сайта та виконаємо реєстрацію облікового запису з логіном “laichuk_artem” та аналогічним паролем.

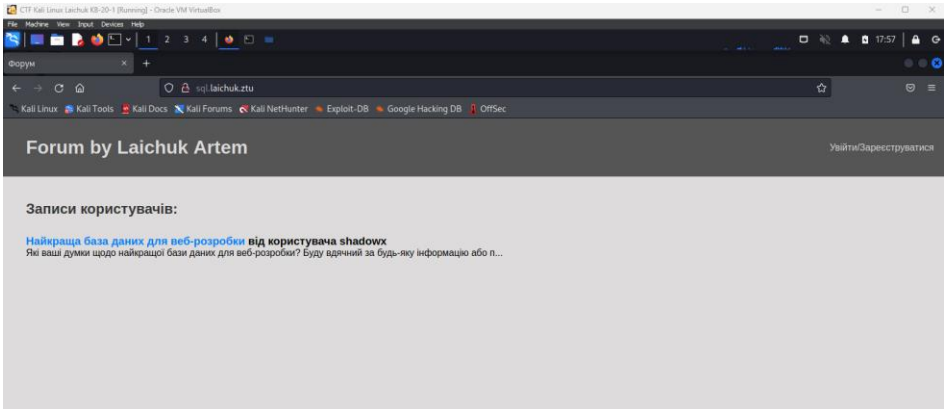


Рисунок 3.5.10 – Головна сторінка веб-сайту

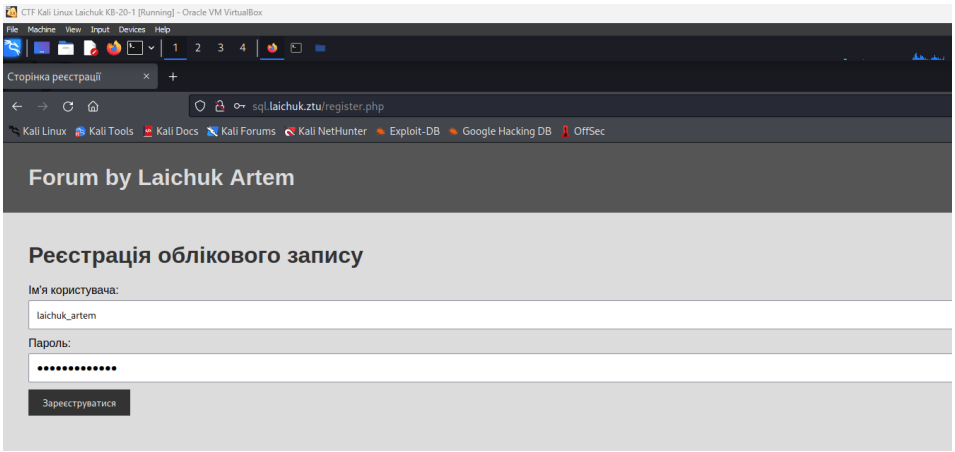


Рисунок 3.5.11 – Процес реєстрації облікового запису

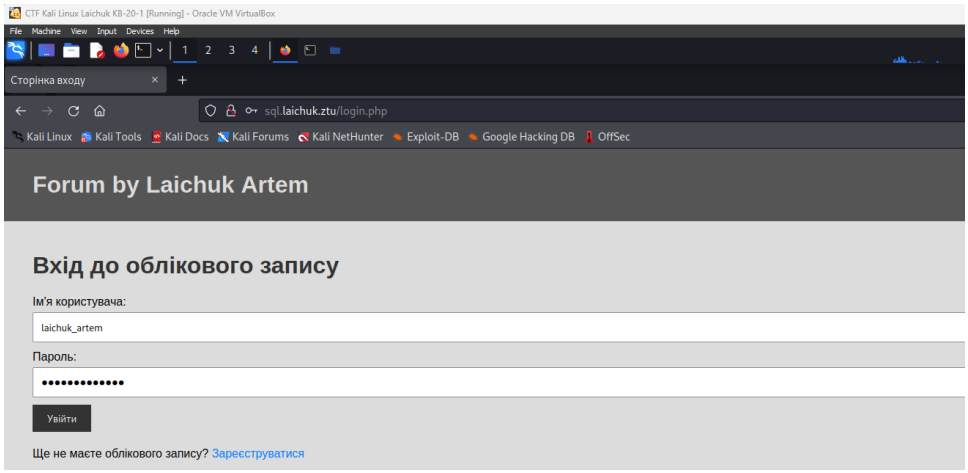


Рисунок 3.5.12 – Вхід до облікового запису після успішної реєстрації

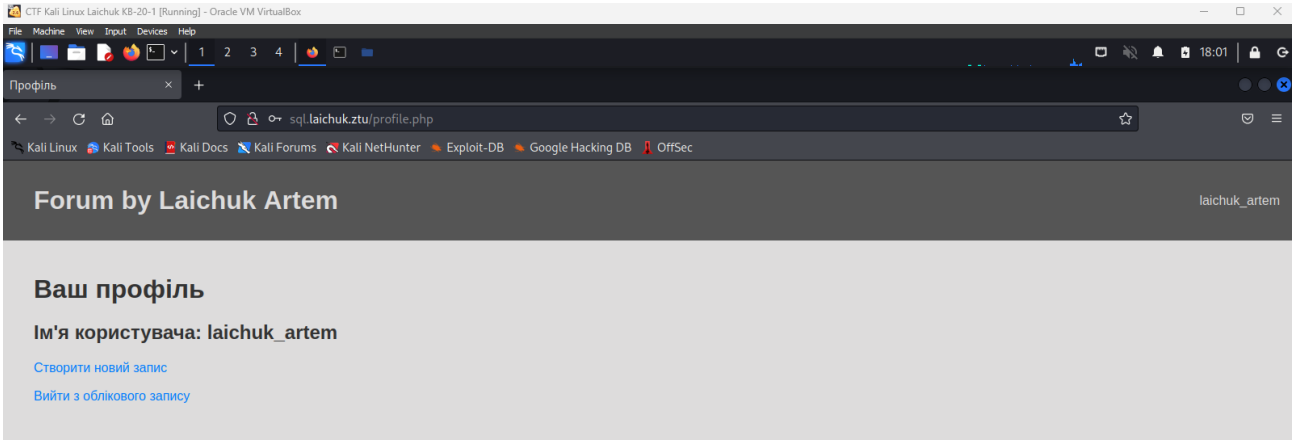


Рисунок 3.5.13 – Успішний вхід до облікового запису

Після успішного входу в обліковий запис (рис. 3.5.13), перейдемо до головної сторінки та натиснемо на завчасно створений запис.

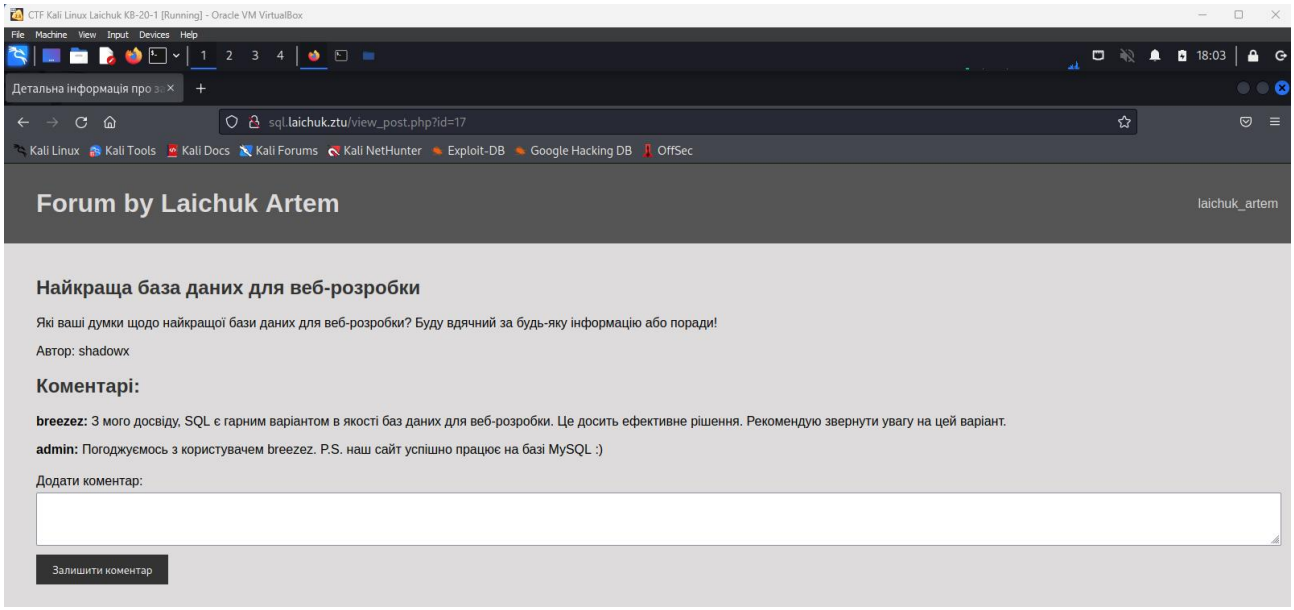


Рисунок 3.5.14 – Детальна інформація про запис

Як бачимо на рис. 3.5.14, на сторінці відображаються коментарі до запису та його id в URL посиланні. Використавши першу підказку, використаємо інструмент “sqlmap” та, для початку, дізнаємось назву бази даних, що використовується на веб-сайті. Для цього використаємо команду “sqlmap -u 'http://sql.laichuk.ztu/view_post.php?id=17' --data="id=17" --current-db”, в яку передамо посилання на цей запис і вразливе поле ID, в яке необхідно здійснити SQL-ін’єкцію.

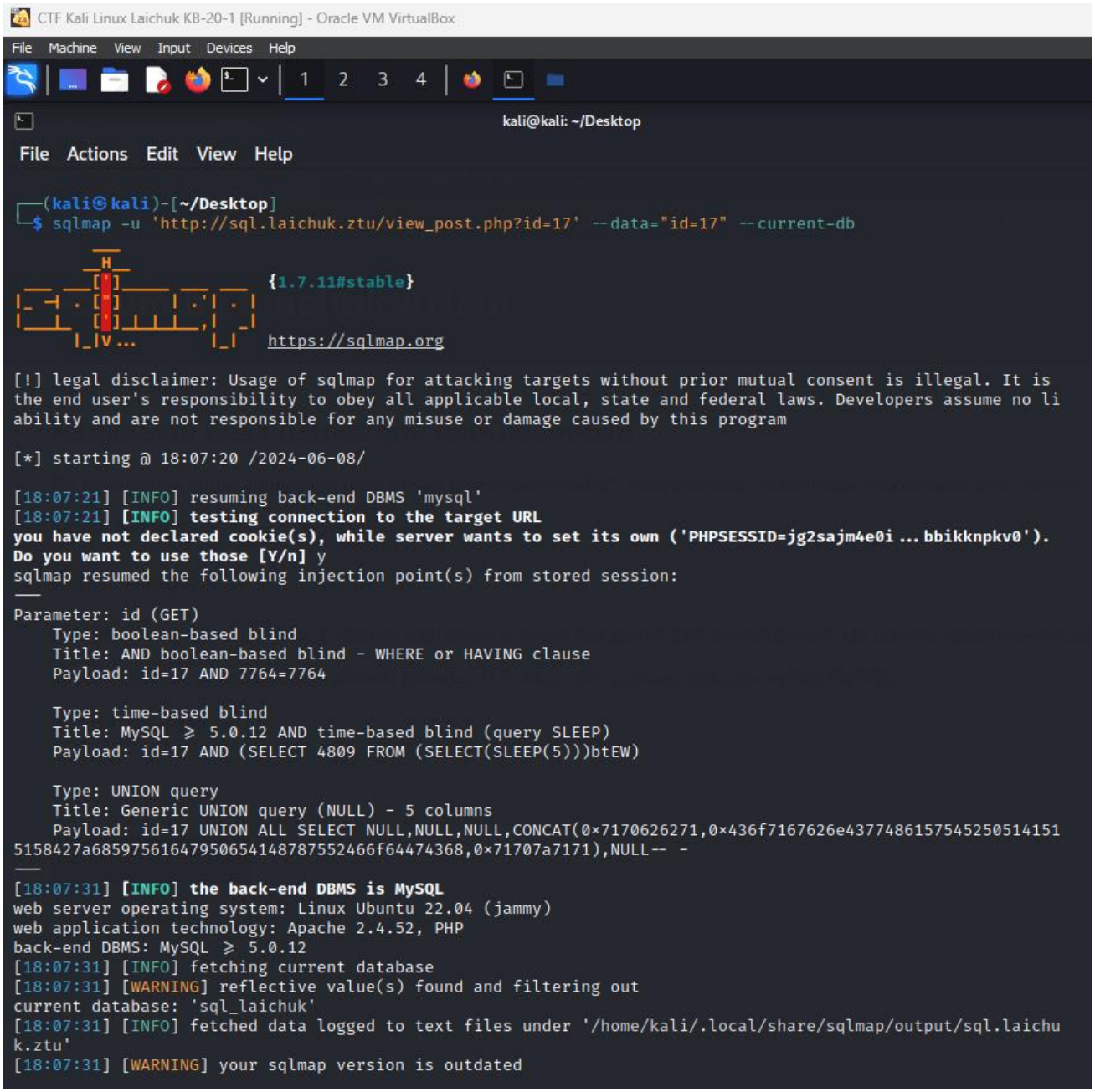


Рисунок 3.5.15 – Успішне виконання SQL-ін’єкції за допомогою інструмента “sqlmap”

На рис. 3.5.15 зображено, що назва бази даних, що використовується на веб-сайті – “sql_laichuk” (current database: ‘sql_laichuk’). Тепер, коли ми знаємо назву бази даних, необхідно дізнатися назви таблиць. Для цього замість параметра “–curent-db” використаємо параметри “-D sql_laichuk --tables”, де вкажемо отриману назву бази даних.

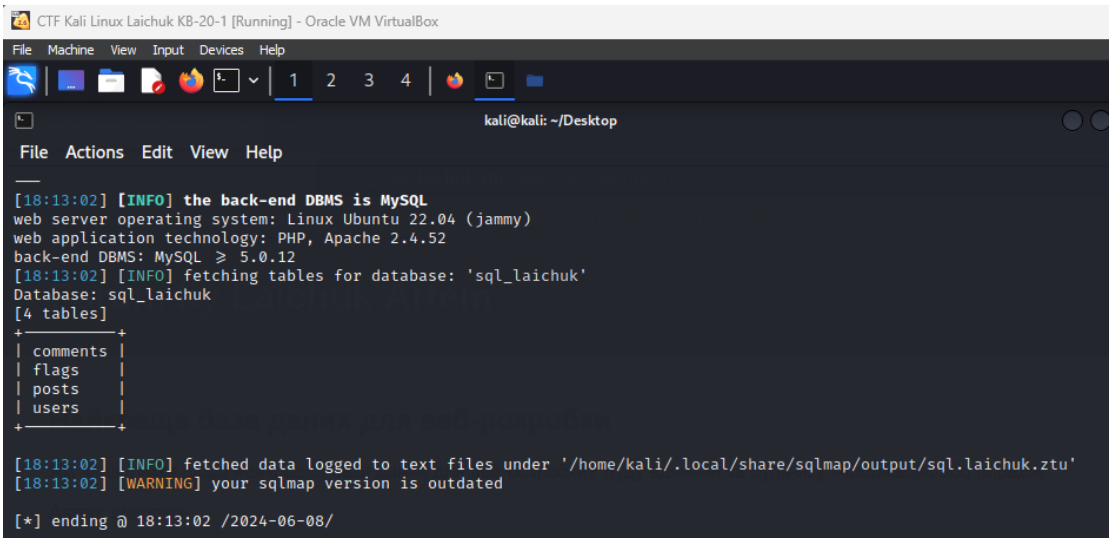


Рисунок 3.5.16 – Успішно отримані назви таблиць бази даних “sql_laichuk”

Наступний крок – отримання даних, що зберігаються в даних таблицях. Для цього використаємо дві команди, в які будуть передаватися назви таблиць “flags” та “users”. У цьому випадку параметром буде “-D sql_laichuk -T flags --dump” (та -T users).

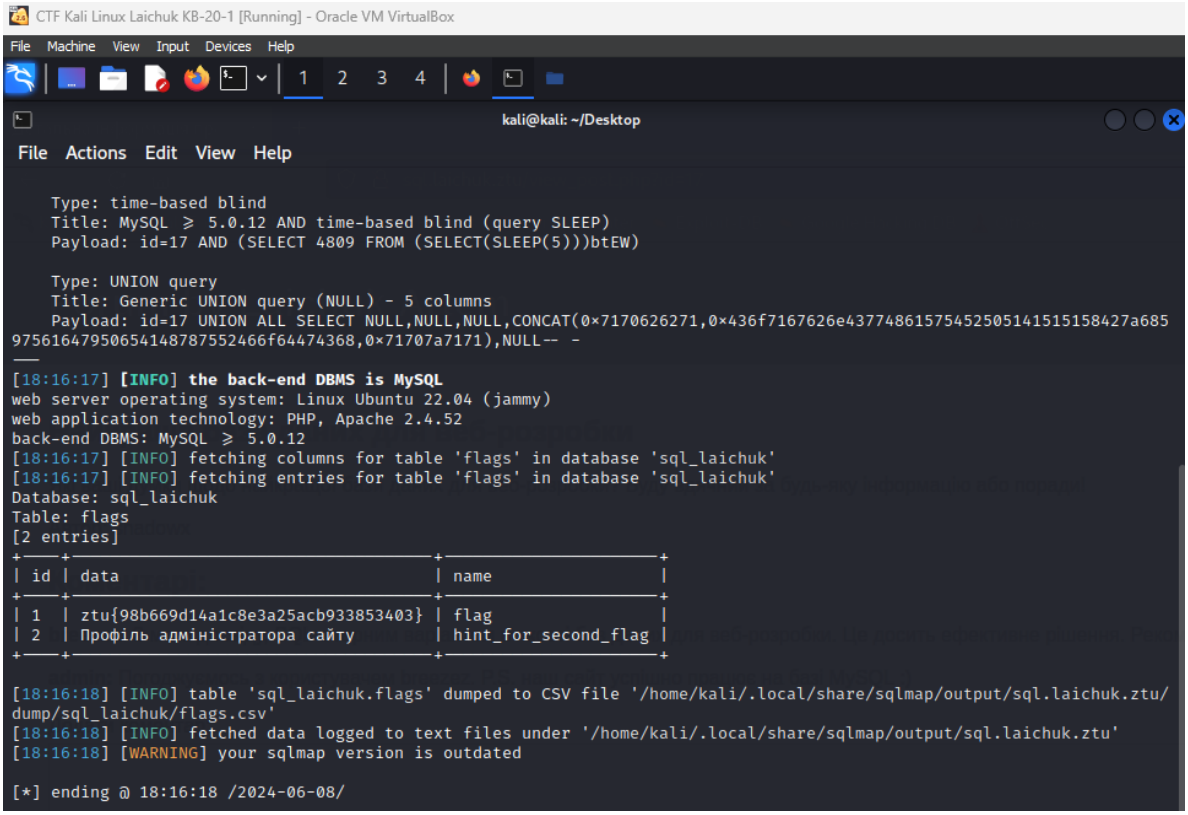


Рисунок 3.5.17 – Успішно отриманий перший прапор із таблиці “flags” за допомогою SQL-ін’єкції з використанням інструмента “sqlmap”

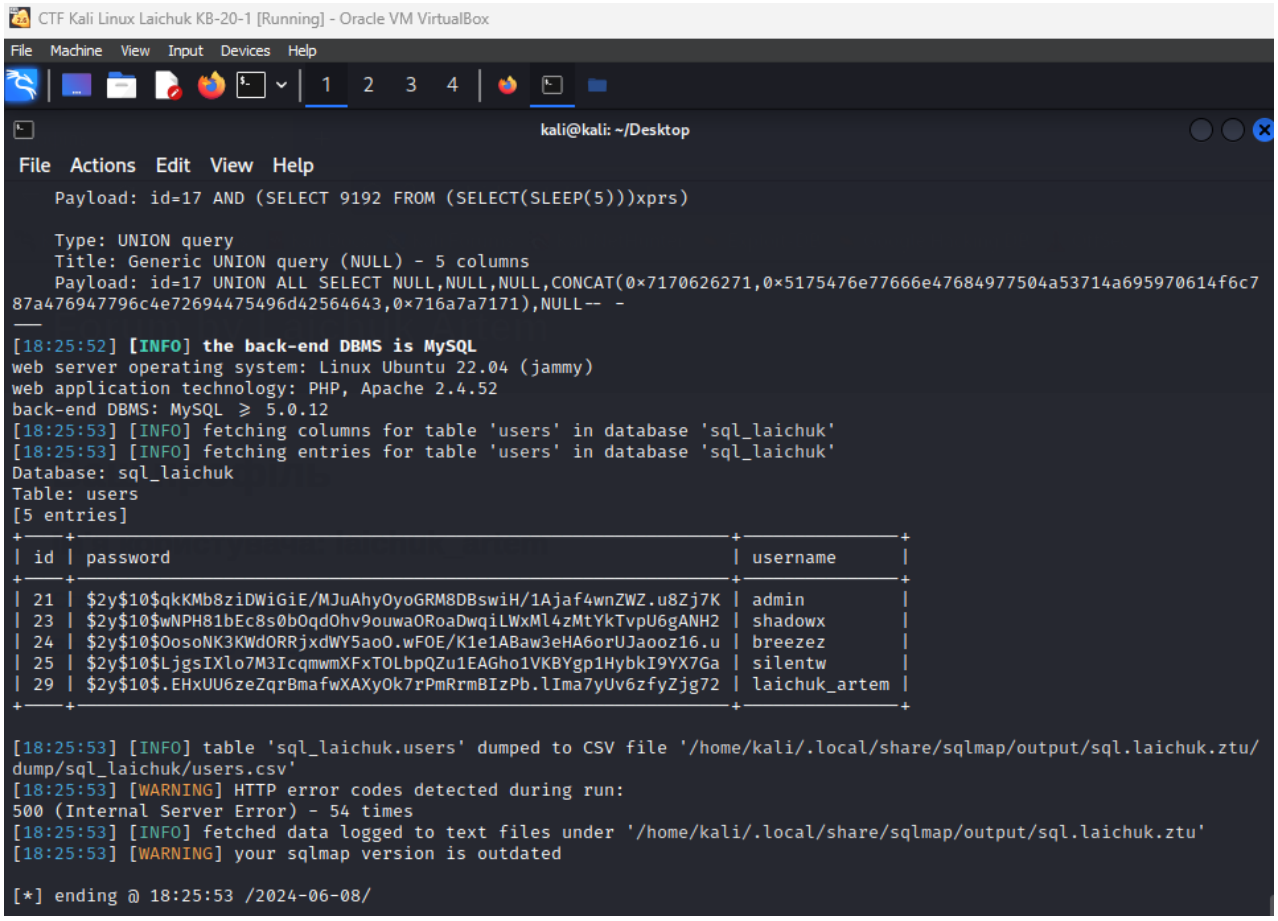


Рисунок 3.5.18 – Успішно отримані дані із таблиці “users” за допомогою SQL-ін’єкції з використанням інструмента “sqlmap”

На даному етапі ми успішно захопили перший прапор, підказку та паролі усіх користувачів. Підказка для захоплення другого прапора вказує на профіль адміністратора веб-сайту, тому необхідно зламати пароль користувача “admin”, що зберігається в таблиці “users”. Для цього скористаємось інструментом “hashcat”.

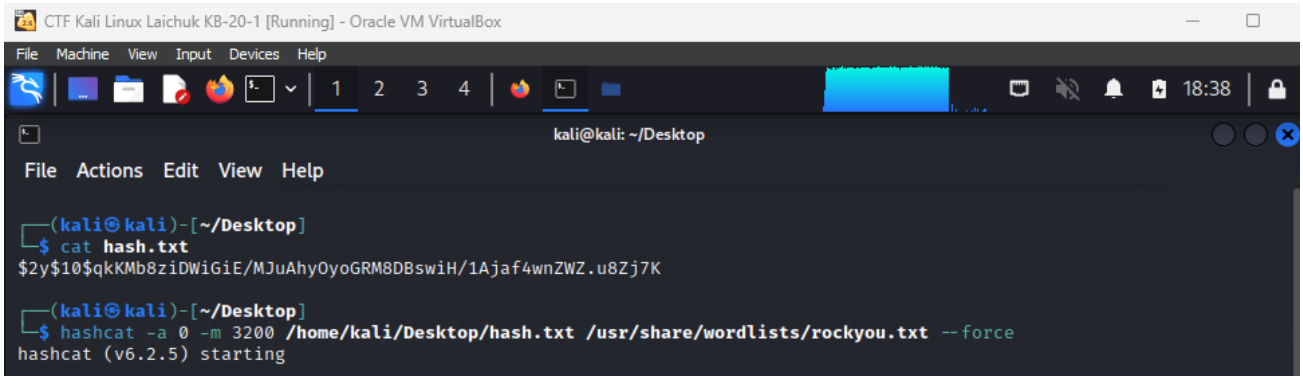


Рисунок 3.5.19 – Початок процесу зламу хеша за допомогою інструмента “hashcat”

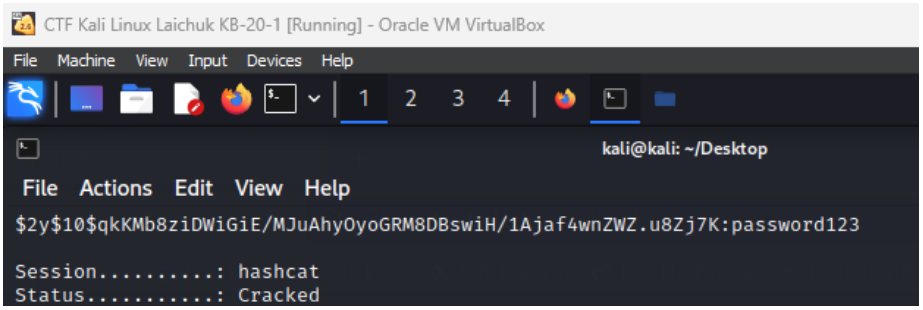


Рисунок 3.5.20 – Успішно зламаний хеш за допомогою інструмента “hashcat”

В результаті успішного зламу хеша (рис. 3.5.20) бачимо, що пароль від облікового запису “admin” – password123. Увійдемо до облікового запису “admin” використовуючи отриманий пароль.

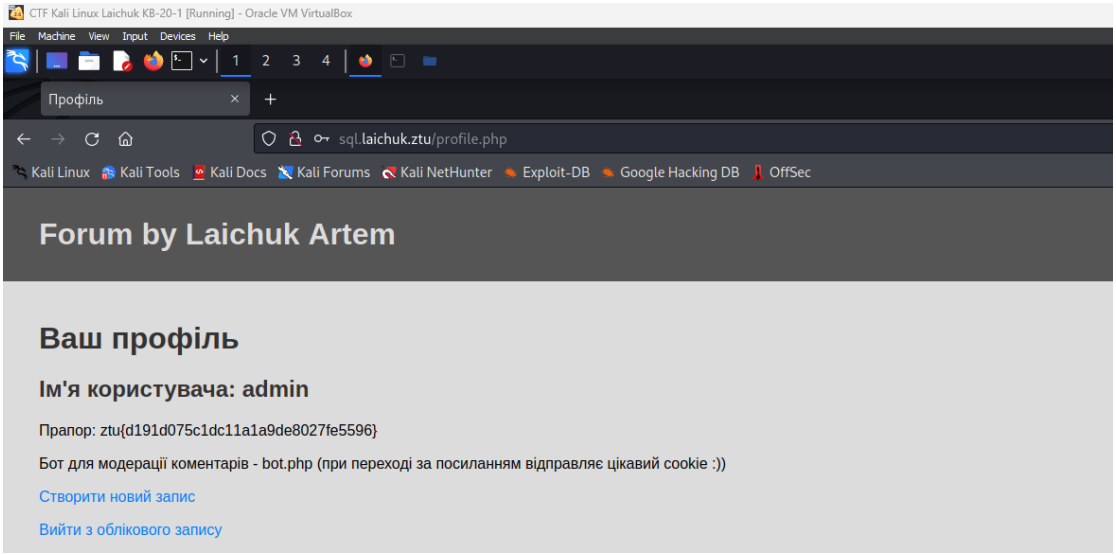


Рисунок 3.5.21 – Успішний вхід до облікового запису “admin”, отримання другого прапора та підказки

Виконавши вхід (рис. 3.5.21), ми отримали другий прапор та наступну підказку. Спробуємо перейти на сторінку bot.php.

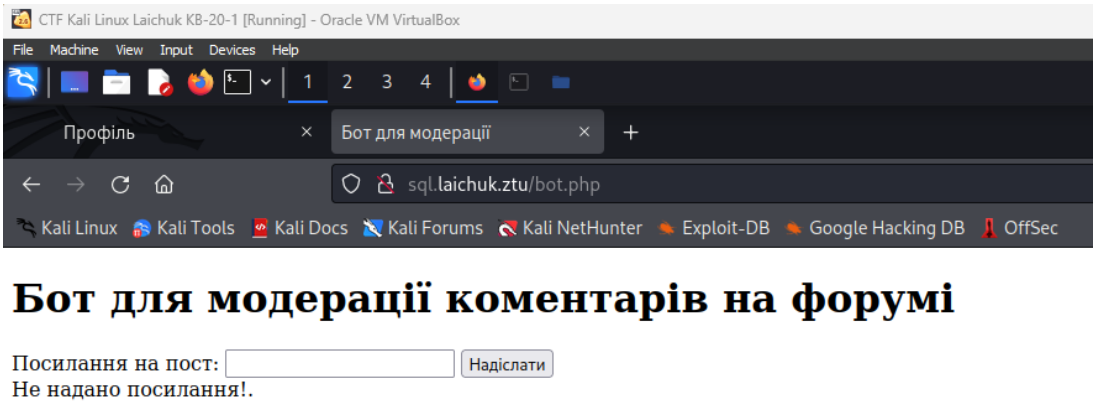


Рисунок 3.5.22 – Сторінка bot.php

Для отримання третього прапору необхідно експлуатувати вразливість XSS. Створимо новий запис на веб-сайті, після чого залишимо коментар із скриптом, який буде перенаправляти cookie кожного, хто перейде до цього запису, на робочу станцію учасника. Спочатку, створимо скрипт на мові PHP, який отримуватиме cookie користувачів та заноситиме їх до текстового файлу.

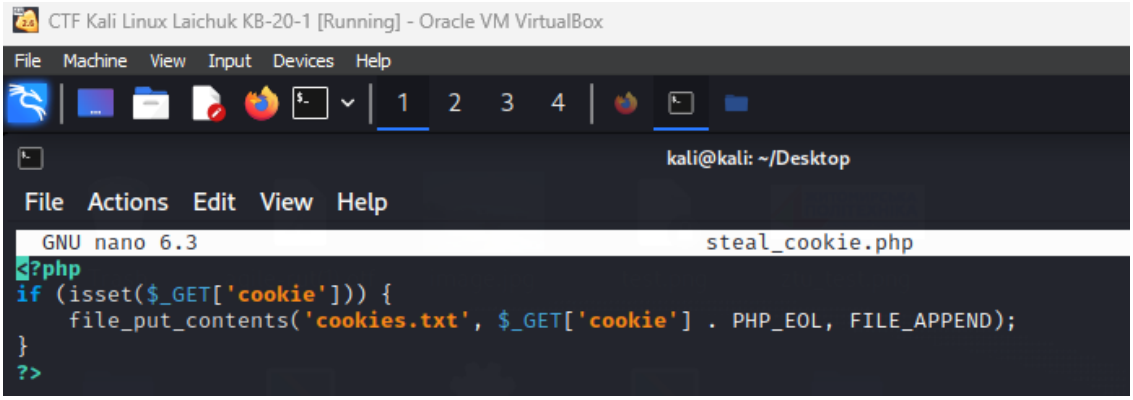


Рисунок 3.5.22 – Скрипт для отримання cookie користувачів

Створимо новий запис на веб-сайті та додамо коментар зі скриптом.

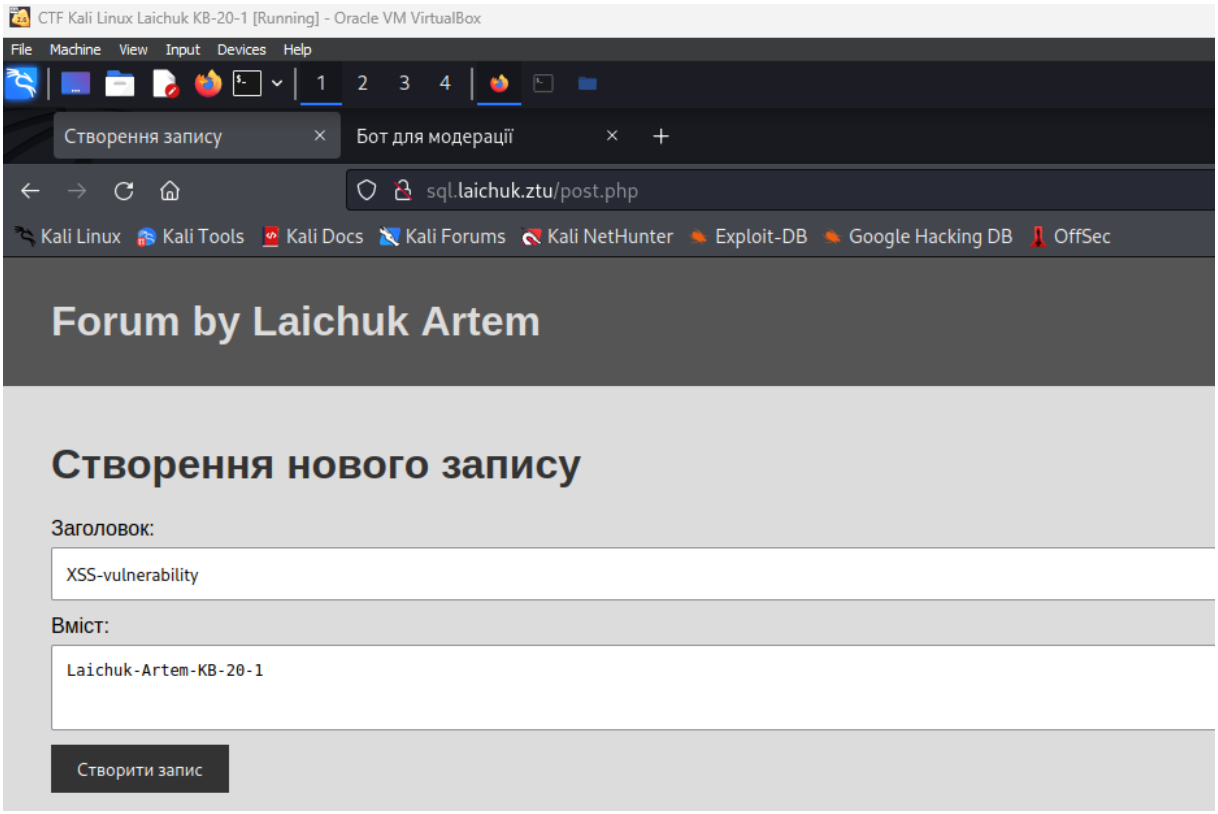


Рисунок 3.5.23 – Процес створення нового запису на веб-сайті

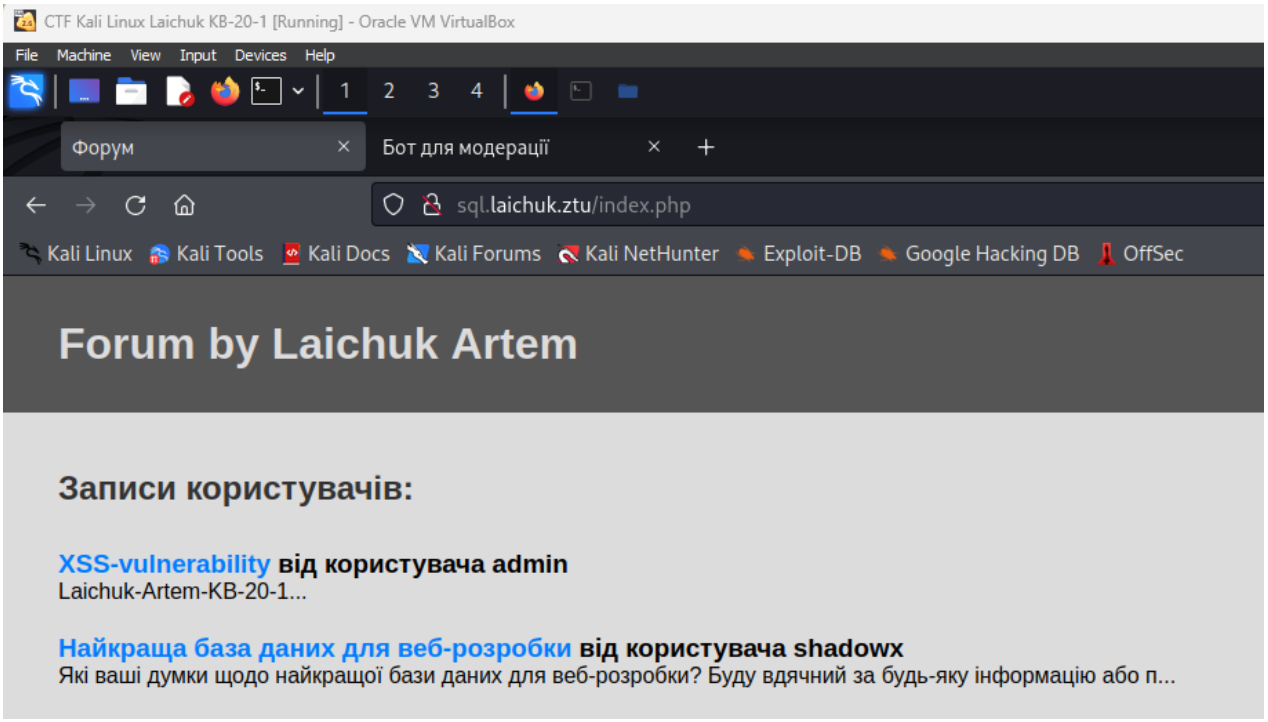


Рисунок 3.5.24 – Успішно створений запис на веб-сайті

Перед додаванням шкідливого коментаря, необхідно запустити вбудований у PHP веб-сервер командою “php -S 201.9.10.2:4444”.

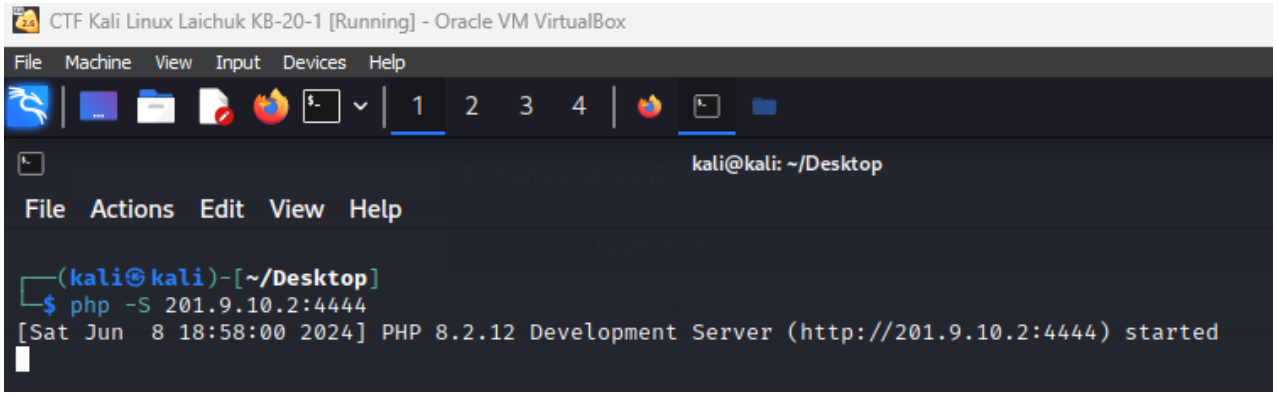


Рисунок 3.5.25 – Успішно запущений веб-сервер

Тепер виконаємо атаку типу XSS (Cross-Site Scripting). Залишимо коментар зі скриптом, який перенаправить будь-кого, хто відкриє даний запис, за посиланням на веб-сервер, що працює на робочій станції учасника.

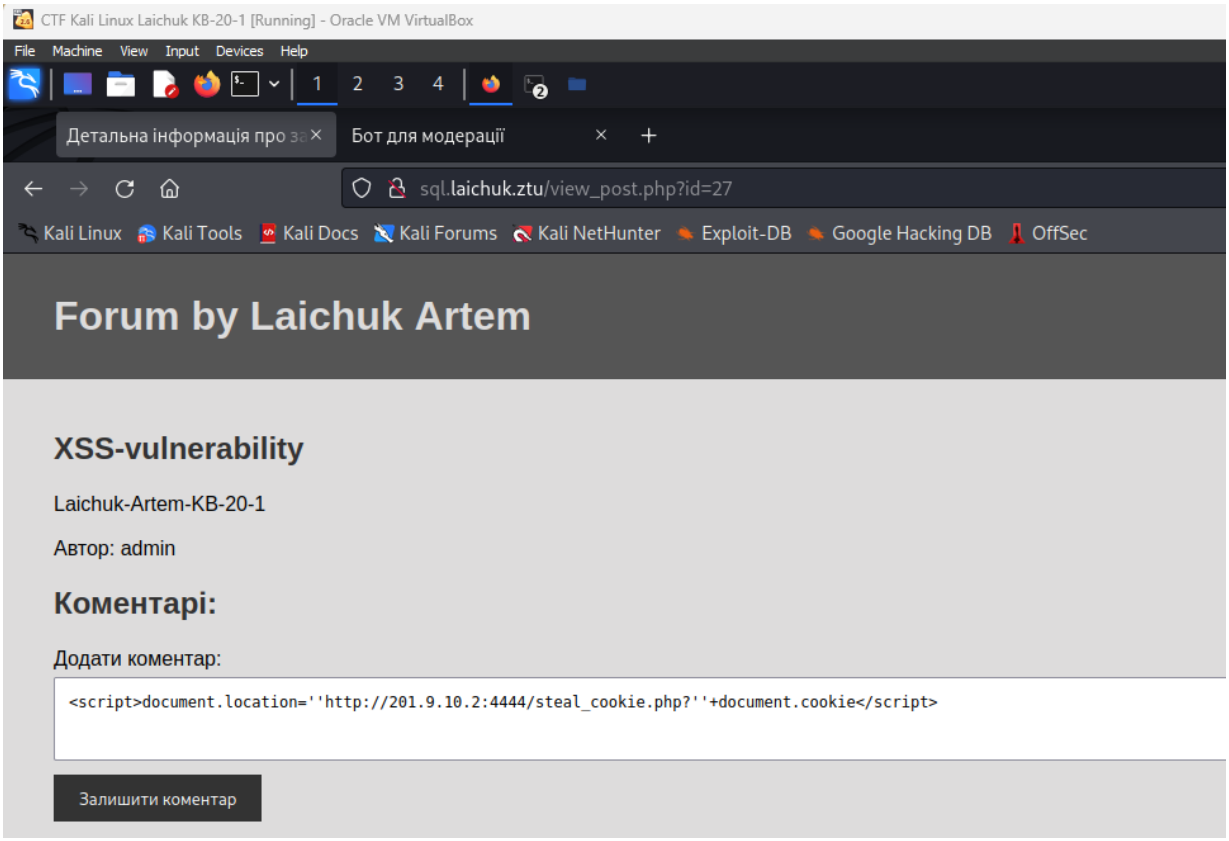


Рисунок 3.5.26 – Процес виконання атаки типу XSS (Cross-Site Scripting)

Після того, як коментар успішно додався на веб-сайт, відправимо в бота посилання на даний пост.

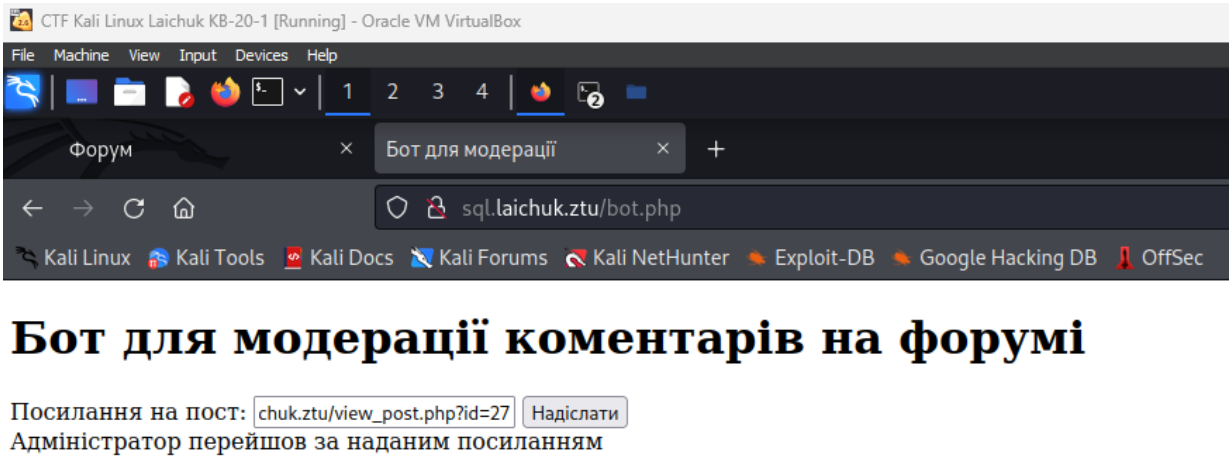


Рисунок 3.5.27 – Успішно відправлена URL-адреса на запис, в якому реалізована атака типу XSS

Фінальним кроком є перевірка веб-сервера на робочій станції учасника на надходження будь-яких запитів та файлу cookies.txt, в який заносяться cookie користувачів.

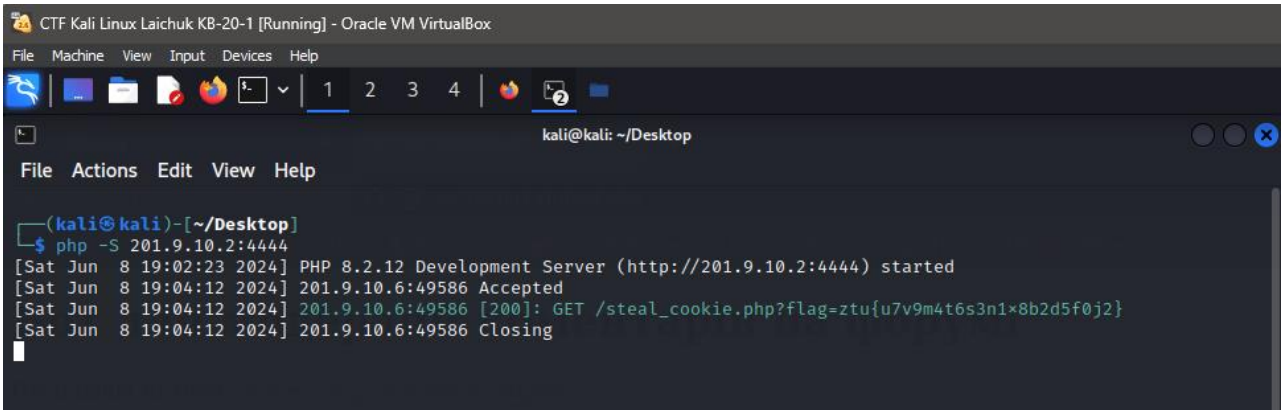


Рисунок 3.5.28 – Успішно отриманий запит на веб-сервер, який містить в собі cookie з третім прапором

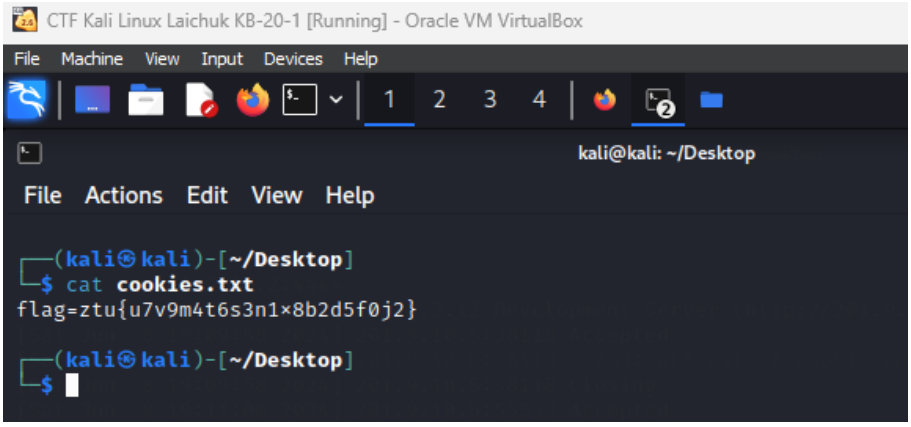


Рисунок 3.5.29 – Успішно занесений cookie до файлу cookies.txt

Як зображено на рис. 3.5.28 - 3.5.29, атака XSS успішно реалізована, в результаті чого третій прапор був успішно захоплений. Загалом, в даному завданні було захоплено три прапора, що свідчить про успішне виконання завдання.

ВИСНОВКИ

В результаті виконання випускної кваліфікаційної роботи бакалавра було успішно досягнуто мети проектування набору завдань з категорії Web для проведення навчальних кібербезпекових досліджень та CTF-змагань з їх подальшими практичною реалізацією, тестуванням та експлуатацією вразливостей.

Детально розглянуто теоретичні аспекти створення веб-додатків та розробки завдань у форматі Capture The Flag. Досліджено архітектуру веб-додатків, включаючи клієнт-серверну модель, фронтенд та бекенд, а також роль веб-серверів та серверів додатків. Розглянуто основи безпеки веб-додатків, включаючи поширені вразливості, методики захисту та інструменти для тестування безпеки. Досліджено рівні складності завдань у контексті CTF-змагань та процес створення завдань, а також їх тестування та валідацію.

Після дослідження процесу створення завдань для навчальних досліджень та CTF-змагань було спроектовано набір із чотирьох завдань та веб-додатків з проведенням аналізу вимог, визначенням цільової аудиторії та контексту застосування, детальним описом кожного завдання та визначенням їх складності. На основі виконаного проектування було обрано інструменти та технології для створення веб-сайтів та реалізації завдань, а саме середовище виконання, вибір мов програмування та веб-фреймворків, вибір та проектування бази даних, а також інструменти, необхідні учасникам для експлуатації вразливостей.

На основі результатів проектування було здійснено практичну реалізацію та тестування створених веб-додатків. Перед початком виконання практичної частини було підготовлено середовище для розробки веб-сайтів, після чого розроблено вебсайт для дослідження пошуку прихованих шляхів та модифікації HTTP-запитів від клієнта до сервера, а також проведено тестування розробленого функціоналу та його експлуатацію для захоплення прапора.Також було створено веб-додатки з вразливостями JSON Web Token, Server-Side Template Injection, CVE-2021-22204, SQLi та Cross-Site Scripting, після чого було проведено тестування розробленого функціоналу та успішну експлуатацію цих вразливостей з подальшим захопленням усіх прапорів.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. 10 web application security best practices. URL: https://medium.com/@Imaginovation/10-web-application-security-best-practices-you-need-to-know-30a6e2fed1a2.
2. All about JWT vulnerabilities. URL: https://bugbase.ai/blog/all-about-jwt-vulnerabilities.
3. Best practices and strategies to secure web applications. URL: https://stfalcon.com/uk/blog/post/building-secure-web-applications-best-practices-and-strategies.
4. Client-Server Model. URL: https://www.geeksforgeeks.org/client-server-model/
5. ExifTool command injection (CVE-2021-22204). URL: https://ine.com/blog/exiftool-command-injection-cve-2021-22204.
6. ExifTool DjVu ANT Perl injection. URL: https://www.rapid7.com/db/modules/exploit/unix/fileformat/exiftool_djvu_ant_perl_injection/.
7. Frontend vs backend. URL: https://www.geeksforgeeks.org/frontend-vs-backend/.
8. How to secure web applications from vulnerabilities. URL: https://mobidev.biz/blog/best-practices-to-secure-web-applications-from-vulnerabilities.
9. How to use hashcat in kali linux. URL: https://thetechdeck.hashnode.dev/how-to-use-hashcat-in-kali-linux.
10. How to use PHP with MYSQL Database. URL: https://www.simplilearn.com/tutorials/php-tutorial/php-with-sql.
11. JWT locations. URL: https://flask-jwt-extended.readthedocs.io/en/stable/token_locations.html#cookies.
12. JWT vulnerabilities (json web tokens). URL: https://book.hacktricks.xyz/pentesting-web/hacking-jwt-json-web-tokens.
13. Khawaja G. Practical web penetration testing: secure web applications using burp suite, nmap, metasploit, and more. Packt Publishing, 2018. 294 p.

14. Najera-Gutierrez G., Ansari J. A. Web penetration testing with kali linux - third edition: explore the methods and tools of ethical hacking with kali linux. Packt Publishing, 2018. 426 p.
15. OWASP top 10. URL: https://owasp.org/Top10/.
16. Pentesting web methodology. URL: https://book.hacktricks.xyz/network-services-pentesting/pentesting-web.
17. Server-side template injection. URL: https://portswigger.net/web-security/server-side-template-injection.
18. SQLmap tutorial. URL: https://hackertarget.com/sqlmap-tutorial/.
19. The difference between front-end vs. back-end. URL: https://www.computerscience.org/bootcamps/resources/frontend-vs-backend/.
20. Web server versus application server. IBM. URL: https://www.ibm.com/topics/web-server-application-server.
21. Why is capture the flag (CTF) important in cyber security?. URL: https://www.eccouncil.org/cybersecurity-exchange/ethical-hacking/capture-the-flag-ctf-cybersecurity/.
22. How to configure Apache2 settings. URL: https://ubuntu.com/server/docs/how-to-configure-apache2-settings.
23. How to configure the Apache web server on an Ubuntu. URL: https://www.digitalocean.com/community/tutorials/how-to-configure-the-apache-web-server-on-an-ubuntu-or-debian-vps.
24. Web servers vs. application server: What's the difference. URL: https://www.openlogic.com/blog/web-server-vs-application-server
25. Top 10 web application vulnerabilities. URL: https://securelist.com/top-10-web-app-vulnerabilities/112144/
26. What is Kali Linux? URL: https://www.kali.org/docs/introduction/what-is-kali-linux/
27. What is Nikto and it's usages? URL: https://www.geeksforgeeks.org/what-is-nikto-and-its-usages/

ДОДАТКИ

Додаток А

Вихідний код веб-сайту Dirb + HTTP Headers

index.php:

```
<!DOCTYPE html>

<html lang="en">
<head>
   <meta charset="UTF-8">
   <title>Laichuk Artem KB-20-1</title>
</head>
<body>
   <div class="container">
      <p>Каталоги, файли, розширення...</p>
   </div>
</body>
</html>
```

flag.php:

```
<?php

$required_headers = [
   'Referer' => 'https://ztu.edu.ua',
   'User-Agent' => 'ZTU',
   'X-Forwarded-For' => '127.0.0.1',
   'Date' => 'Mon, 20 May 2024 10:00:00 GMT+3',
];

$missing_headers = [];
$headers_ok = true;

foreach ($required_headers as $header => $value) {
   $server_key = 'HTTP_' . strtoupper(str_replace('-', '_', $header));
   if ($header === 'Date') {
      $server_key = 'HTTP_DATE';
   }
   if (!isset($_SERVER[$server_key]) || $_SERVER[$server_key] !== $value) {
      $headers_ok = false;
      $missing_headers[$header] = $value;
      break;
   }
}

if ($headers_ok) {
   $flag = file_get_contents('s3cr3t_f1l3.txt');
if (isset($_SERVER['HTTP_RANGE'])) {
   preg_match('/^bytes=(\d+)-(\d+)$/', $_SERVER['HTTP_RANGE'], $matches);

   $start = intval($matches[1]);
   $end = intval($matches[2]);

   if ($end >= strlen($flag)) {
      $end = strlen($flag) - 1;
   }

   header('Content-Type: text/plain');
   header('Content-Length: ' . ($end - $start + 1));
   echo substr($flag, $start, $end - $start + 1);
} else {
   $flag_part = substr($flag, 0, 6);
   echo "<p>$flag_part...</p>";
```

```
    echo "<p>Недостатній діапазон для отримання повного прапора!</p>";
}
} else {
    echo "<h1>Доступ заборонений</h1>";
    echo "<p>Прапор можна отримати перейшовши з офіційного сайту 'Житомирської політехніки' за допомогою браузера 'ZTU', використовуючи локальну адресу, лише 20 травня 2024 року о 10:00:00</p>";

    foreach ($missing_headers as $header => $value) {
        switch ($header) {
            case 'Referer':
                echo "<p>Відмовлено у доступі: Ви перейшли не з офіційного сайту 'Житомирської політехніки'</p>";
                break;
            case 'User-Agent':
                echo "<p>Відмовлено у доступі: Некоректний браузер.</p>";
                break;
            case 'X-Forwarded-For':
                echo "<p>Відмовлено у доступі: IP-адреса не є локальною.</p>";
                break;
            case 'Date':
                echo "<p>Відмовлено у доступі: Некоректні дата та час.</p>";
                break;
        }
        break;
    }

}
?>
```

Додаток Б

Вихідний код веб-сайту JWT + SSTI

app.py:

```
from flask import Flask, render_template, render_template_string, request, jsonify, redirect, url_for, make_response, send_from_directory, flash, abort
import jwt
from jwt.exceptions import DecodeError
import os
import random
import string
from jinja2 import Template, Environment

app = Flask(__name__)

# Встановлюємо секретний ключ для JWT-токена
app.config['SECRET_KEY'] = 'i1U2We3THEY4!'
app.root_path = '/var/www/jwt_ssti/'

users = {'hint': {'password': '0J3QtdGW0YHQvdGD0Y7Rh9C40Lkg0LrQvtGA0LjRgdGC0YPQstCw0Yc7IEppbmphMg', 'role': 'guest'}}

# Функція для генерації унікального та випадкового прапора для користучава
def generate_flag():
    return 'ztu{' + ''.join(random.choice(string.ascii_lowercase + string.digits) for _ in range(25)) + '}'

# Генерація першого прапору
flag = generate_flag()

# Додаємо перший прапор до словника користувача
users['flag №1'] = {'password': flag, 'role': 'guest'}

env = Environment()
env.globals.update(os=os)
env.globals.update(url_for=url_for)
project_dir = os.path.dirname(os.path.abspath(__file__))
# Шлях до головної сторінки
@app.route('/')
def index():
    return render_template('index.html')

# Реалізація функції входу в обліковий запис із створенням та встановленням в cookie токена JWT
@app.route('/login', methods=['GET', 'POST'])
def login():
    if request.method == 'POST':
        username = request.form.get('username')
        password = request.form.get('password')
        if username in users and users[username]['password'] == password:
            role = users[username]['role']
            token = jwt.encode({'username': username, 'role': role}, app.config['SECRET_KEY'], algorithm='HS256')
            response = make_response(redirect(url_for('profile')))
            response.set_cookie('access_token', token)
            return response
        return jsonify({"msg": "Неправильні дані для входу"}), 401
    return render_template('login.html')

# Реалізація функції реєстрації облікового запису із створенням та встановленням в cookie токена JWT
@app.route('/register', methods=['GET', 'POST'])
def register():
    if request.method == 'POST':
        username = request.form.get('username')
        password = request.form.get('password')
        if username not in users:
            users[username] = {'password': password, 'role': 'guest'}
            token = jwt.encode({'username': username, 'role': 'guest'}, app.config['SECRET_KEY'], algorithm='HS256')
            response = make_response(redirect(url_for('profile')))
            response.set_cookie('access_token', token)
```

```
            return response
        return jsonify({"msg": "Користувач вже існує"}), 400
    return render_template('register.html')

# Шлях до профілю облікового запису користувача
@app.route('/profile')
def profile():
    access_token = request.cookies.get('access_token')
    if access_token:
        try:
            token_data = jwt.decode(access_token, app.config['SECRET_KEY'], algorithms=['HS256'])
            username = token_data['username']
            role = token_data['role']
            return render_template('profile.html', username=username, role=role)
        except DecodeError:
            return jsonify({"error": "Помилка авторизації: неправильний токен"}), 401
        except jwt.ExpiredSignatureError:
            return jsonify({"error": "Помилка авторизації: термін дії токена закінчився"}), 401
    else:
        return jsonify({"error": "Помилка авторизації: відсутній токен"}), 401

# Реалізація виходу з облікового запису та очищення cookie на веб-сайті
@app.route('/logout', methods=['POST'])
def logout():
    response = make_response(redirect(url_for('index')))
    response.set_cookie('access_token', expires=0)
    return response

# Шлях до панелі адміністратора
@app.route('/adm1n_pan3l')
def admin_panel():
    access_token = request.cookies.get('access_token')
    if access_token:
        try:
            token_data = jwt.decode(access_token, app.config['SECRET_KEY'], algorithms=['HS256'])
            role = token_data.get('role')
            # Якщо роль облікового запису - admin, генерується випадковий прапор та заноситься до файлу flag2.txt
            if role == 'admin':
                flag = generate_flag()
                flag_path = os.path.join(project_dir, 'flag2.txt')
                with open(flag_path, 'w') as flag_file:
                    flag_file.write(flag)
                return render_template('adm1n_pan3l.html', users=users)
            else:
                return jsonify({"msg": "Доступ до цієї сторінки заборонений"}), 403
        except jwt.ExpiredSignatureError:
            return jsonify({"msg": "Термін дії токена закінчився"}), 401
    else:
        return jsonify({"msg": "Відсутній заголовок авторизації"}), 401
# Функція для пошуку користувачів
@app.route('/search_user', methods=['POST'])
def search_user():
    username = request.form.get('username')
    if not username:  # Якщо рядок пошуку порожній, виводимо в таблицю всіх користувачів
        return render_template('adm1n_pan3l.html', users=users, user_found=None)
    else:
        user_found = users.get(username)
        if user_found:
            return render_template('adm1n_pan3l.html', users=users, user_found=username)
        else:
            # Використання вразливого до атаки SSTI render_template_string без належної фільтрації введених користувачем значень
            return render_template_string("<p>Користувача з іменем: %s не існує</p><a href='/adm1n_pan3l'>Повернутися до панелі
адміністратора</a><br/>" % username)
if __name__ == '__main__':
    app.run(debug=True)
```

index.html:

```
<!DOCTYPE html>
<html lang="en">
```

```
<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <title>Головна сторінка</title>
   <link rel="stylesheet" href="{{ url_for('static', filename='styles.css') }}">
</head>
<body>
   <header>
      <div class="header-content">
         <h1>Завдання №1</h1>
         {% if username %}
            <div class="login-link">
               <a href="{{ url_for('profile') }}">{{ username }}</a>
            </div>
            {% if role == 'admin' %}
               <div class="login-link">
                  <a href="{{ url_for('admin_panel') }}">Admin Panel</a>
               </div>
            {% endif %}
         {% else %}
            <div class="login-link">
               <a href="/login">Увійти</a>
            </div>
         {% endif %}
      </div>
   </header>

   <main>
      <div class="main-content">
         <h2>Головна сторінка сайту</h2>


<p>На даному веб-сайті необхідно використати дві вразливості для отримання двох прапорів.</p>
      </div>
   </main>
   <footer>
      <div class="footer-content">
         <p>Made by Artem Laichuk</p>
      </div>
   </footer>

</body>
</html>
```

login.html:

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <link rel="stylesheet" type="text/css" href="{{ url_for('static', filename='styles.css') }}">
   <title>Сторінка входу</title>
</head>
<body>
   <div class="login-container">
      <h2>Вхід до облікового запису</h2>
      <form id="loginForm" action="/login" method="post">
         <input type="text" name="username" placeholder="Логін" required>
         <input type="password" name="password" placeholder="Пароль" required>
         <button type="submit">Увійти</button>
      </form>
      <p>Немає облікового запису? <a href="/register">Зареєструватися</a></p>
   </div>
</body>
</html>
```

profile.html:

```
<!DOCTYPE html>
<html lang="en">
```

```
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Профіль</title>
    <link rel="stylesheet" href="{{ url_for('static', filename='styles.css') }}">
</head>
<body>
    <header>
        <div class="header-content">
            <h1>Профіль користувача</h1>
            {% if username %}
                {% if role == 'admin' %}
                    <div class="login-link">
                        <a href="{{ url_for('admin_panel') }}">Панель адміністратора</a>
                    </div>
                {% endif %}
            {% endif %}
        </div>
    </header>

    <div class="profile-container">
        {% if role == 'guest' %}
            <p>Для доступу до панелі адміністратора, необхідно увійти до облікового запису з роллю "admin"</p>
        {% endif %}

        <p>Ім'я облікового запису: {{ username }}</p>
        <p>Роль: {{ role }}</p>

        {% with messages = get_flashed_messages() %}
            {% if messages %}
                <script>
                    alert("{{ messages[0] }}");
                </script>
            {% endif %}
        {% endwith %}

        <form class="logout" action="{{ url_for('logout') }}" method="post">
            <button type="submit">Вийти з облікового запису</button>
        </form>
    </div>

    <footer>
        <div class="footer-content">
            <p>Made by Artem Laichuk</p>
        </div>
    </footer>
</body>
</html>
```

register.html:

```
<!DOCTYPE html>
<html lang="en">
    <head>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
        <title>Реєстрація</title>
        <link rel="stylesheet" href="{{ url_for('static', filename='styles.css') }}">
    </head>
<body>
    <div class="register-container">
        <h2>Реєстрація облікового запису</h2>
        <form action="/register" method="POST">
            <input type="text" name="username" placeholder="Логін" required>
            <input type="password" name="password" placeholder="Пароль" required>
            <button type="submit">Зареєструватися</button>
        </form>
    </div>
</body>
</html>
```

adm1n_pan3l.html:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Панель адміністратора</title>
    <link rel="stylesheet" href="{{ url_for('static', filename='styles.css') }}">
</head>
<body>
    <header>
        <div class="header-content">
            <h1>Панель адміністратора</h1>
            <div class="login-link">
                <a href="{{ url_for('profile') }}">Профіль користувача</a>
            </div>
        </div>

    </header>
    <form class="search-form" action="{{ url_for('search_user') }}" method="post">
        <input type="text" name="username" placeholder="Введіть ім'я облікового запису для пошуку...">
        <button type="submit">Пошук</button>
    </form>
    <table>
        <tr>
            <th>Username</th>
            <th>Password</th>
        </tr>
        {% if user_found %}
            <tr>
                <td>{{ user_found }}</td>
                <td>{{ users[user_found].password }}</td>
            </tr>
        {% else %}
            {% for user, data in users.items() %}
                <tr>
                    <td>{{ user }}</td>
                    <td>{{ data.password }}</td>
                </tr>
            {% endfor %}
        {% endif %}
    </table>
    <footer>
        <div class="footer-content">
            <p>Made by Artem Laichuk</p>
        </div>
    </footer>
</body>
</html>
```

Додаток В

Вихідний код веб-сайту ExifTool-Online

index.php:

```
<!DOCTYPE html>

<html lang="en">
<head>
   <meta charset="UTF-8">
   <title>Exiftool Online</title>
   <link rel="stylesheet" href="static/styles.css">
</head>
<body>
   <header>
      <h1>Exiftool Online</h1>
      <p>CyberRangers Laichuk Artem</p>
   </header>
   <h1>Виведення EXIF-даних зображення</h1>
   <p>На даному веб-сайті необхідно отримати один прапор</p>
   <form action="upload.php" method="post" enctype="multipart/form-data">
      Оберіть зображення для завантаження:
      <input type="file" name="fileToUpload" id="fileToUpload">
      <input type="submit" value="Завантажити" name="submit">
   </form>
</body>
</html>
```

upload.php:

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <title>Upload Image</title>
   <link rel="stylesheet" href="static/styles_upload.css">
</head>
<body>
   <header>
      <h1>Exiftool Online</h1>
      <p>CyberRangers Laichuk Artem</p>
   </header>
   <div class="container">
      <?php
      $target_dir = "./uploads/";

      $file_name = basename($_FILES["fileToUpload"]["name"]);
      $file_name = preg_replace('/[^A-Za-z0-9_\-\.]/', '_', $file_name);
      $target_file = $target_dir . $file_name;
      $uploadOk = 1;
      $imageFileType = strtolower(pathinfo($target_file, PATHINFO_EXTENSION));

      if ($imageFileType != "jpg" && $imageFileType != "jpeg" && $imageFileType != "png") {
         echo "Дозволені зображення лише в форматах JPG, JPEG, PNG.";
         $uploadOk = 0;
      }

      if ($uploadOk == 1 && move_uploaded_file($_FILES["fileToUpload"]["tmp_name"], $target_file)) {
         echo "Файл " . htmlspecialchars($file_name) . " успішно завантажений.<br><br>";

         $uploaded_image = htmlspecialchars($file_name);
         echo "<div style='text-align:center;'><img src='uploads/$uploaded_image' alt='Uploaded Image' style='max-width:
500px;'></div><br><br>";
      } else {
         echo "Виникла помилка під час завантаження зображення";
      }

      if ($uploadOk == 1) {
```

```
        echo "<h2>EXIF-дані:</h2>";
        $escaped_target_file = escapeshellarg($target_file);
        $exif_output = shell_exec("exiftool $escaped_target_file");
        $exif_lines = explode("\n", trim($exif_output));

        echo "<div class='exif-columns'>";
        foreach ($exif_lines as $line) {
            echo htmlspecialchars($line) . "<br>";
        }
        echo "</div>";
    }

    echo "<br><div class='back-link'><a href='index.php'>Повернутись на головну сторінку</a></div>";
    ?>
  </div>
</body>
</html>
```

Додаток Г

Вихідний код веб-сайту SQL + XSS

db.php:

```
<?php
$servername = "localhost";
$user = "laichuk_user";
$pass = "123456LKJhgf";
$dbname = "sql_laichuk";

// Створюємо підключення з базою даних
$conn = new mysqli($servername, $user, $pass, $dbname);

// Перевірка підключення з базою даних
if ($conn->connect_error) {
   die("Connection failed: " . $conn->connect_error);
}
?>
```

register.php:

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <title>Сторінка реєстрації</title>
   <link rel="stylesheet" href="/static/styles.css">
</head>
<body>
   <header>
      <div class="header-left">
         <h1>Forum by Laichuk Artem</h1>
      </div>
      <div class="header-right">
         <a class="header-link" href="login.php">Увійти/Зареєструватися</a>
      </div>
   </header>
   <main>
      <h1>Реєстрація облікового запису</h1>
      <form method="post" action="">
         <label for="username">Ім'я користувача:</label>
         <input type="text" id="username" name="username" required><br>
         <label for="password">Пароль:</label>
         <input type="password" id="password" name="password" required><br>
         <input type="submit" value="Зареєструватися">
      </form>
      <?php
      if ($_SERVER["REQUEST_METHOD"] == "POST") {
         $username = $_POST['username'];
         $password = password_hash($_POST['password'], PASSWORD_BCRYPT); // Хешування пароля за допомогою bcrypt

         include 'db.php';

         $stmt = $conn->prepare("INSERT INTO users (username, password) VALUES (?, ?)");
         $stmt->bind_param("ss", $username, $password);
         if ($stmt->execute()) {
            echo "Реєстрація пройшла успішно! <a href='login.php'>Перейти до входу в обліковий запис</a>";
         } else {
            echo "Error: " . $stmt->error;
         }
         $stmt->close();
         $conn->close();
      }
      ?>
   </main>
</body>
</html>
```

login.php

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <title>Сторінка входу</title>
   <link rel="stylesheet" href="/static/styles.css">
</head>
<body>
   <header>
      <div class="header-left">
         <h1>Forum by Laichuk Artem</h1>
      </div>
      <div class="header-right">
         <?php
         session_start();
         if (isset($_SESSION['username'])) {
            echo "<a class='header-link' href='profile.php'>" . $_SESSION['username'] . "</a>";
         } else {
            echo "<a class='header-link' href='login.php'>Увійти/Зареєструватися</a>";
         }
         ?>
      </div>
   </header>
   <main>
      <h1>Вхід до облікового запису</h1>
      <form method="post" action="">
         <label for="username">Ім'я користувача:</label>
         <input type="text" id="username" name="username" required><br>
         <label for="password">Пароль:</label>
         <input type="password" id="password" name="password" required><br>
         <input type="submit" value="Увійти">
      </form>
      <p>Ще не маєте облікового запису? <a class="body-link" href="register.php">Зареєструватися</a></p>
      <?php
      if ($_SERVER["REQUEST_METHOD"] == "POST") {
         $username = $_POST['username'];
         $password = $_POST['password'];

         include 'db.php';

         $stmt = $conn->prepare("SELECT id, username, password FROM users WHERE username = ?");
         $stmt->bind_param("s", $username);
         $stmt->execute();
         $result = $stmt->get_result();

         if ($result->num_rows > 0) {
            $row = $result->fetch_assoc();
            if (password_verify($password, $row['password'])) {
               session_start();
               $_SESSION['username'] = $row['username'];
               $_SESSION['user_id'] = $row['id'];
               header("Location: profile.php");
               exit();
            } else {
               echo "Невірний логін або пароль!";
            }
         } else {
            echo "Невірний логін або пароль!";
         }
         $stmt->close();
         $conn->close();
      }
      ?>
   </main>
</body>
</html>
```

index.php:

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <title>Форум</title>
   <link rel="stylesheet" href="/static/styles.css">
</head>
<body>
   <header>
      <div class="header-left">
         <h1>Forum by Laichuk Artem</h1>
      </div>
      <div class="header-right">
         <?php
         session_start();
         if (isset($_SESSION['username'])) {
            echo "<a class='header-link' href='profile.php'>" . $_SESSION['username'] . "</a>";
         } else {
            echo "<a class='header-link' href='login.php'>Увійти/Зареєструватися</a>";
         }
         ?>
      </div>
   </header>
   <main>
      <h2>Записи користувачів:</h2>
      <?php
      ini_set('display_errors', 1);
      ini_set('display_startup_errors', 1);
      error_reporting(E_ALL);

      include 'db.php';

      $result = $conn->query("SELECT posts.id, posts.title, posts.content, users.username
                  FROM posts
                  JOIN users ON posts.author_id = users.id
                  ORDER BY posts.id DESC");
      if ($result->num_rows > 0) {
         while($row = $result->fetch_assoc()) {
            echo "<div class='post'>";
            echo "<h3><a class='body-link' href='view_post.php?id=".$row['id']."'>".$row['title']."</a> від користувача
".$row['username']."</h3>";
            echo "<p>".mb_substr($row['content'], 0, 100, 'UTF-8')."...</p>";
            echo "</div>";
         }
      } else {
         echo "Не знайдено жодного поста.";
      }
      $conn->close();
      ?>
   </main>
</body>
</html>
```

profile.php:

```
<?php
session_start();
$flag_env_name = 'ADMIN_FLAG';

if (!isset($_SESSION[$flag_env_name])) {
   $flag = 'ztu{' . bin2hex(random_bytes(15)) . '}';
   $_SESSION[$flag_env_name] = $flag;
} else {
   $flag = $_SESSION[$flag_env_name];
}
?>
<!DOCTYPE html>
<html lang="en">
<head>
```

```
    <meta charset="UTF-8">
    <title>Профіль</title>
    <link rel="stylesheet" href="/static/styles.css">
</head>
<body>
    <header>
        <div class="header-left">
            <h1><a class='header-link' href="index.php">Forum by Laichuk Artem</a></h1>
        </div>
        <div class="header-right">
            <?php
            if (!isset($_SESSION['username'])) {
                header("Location: login.php");
                exit();
            }
            $username = $_SESSION['username'];
            echo "<a class='header-link' href='profile.php'>" . $_SESSION['username'] . "</a>";
            ?>
        </div>
    </header>
    <main>
        <h1>Ваш профіль</h1>
        <?php
        echo "<h2>Ім'я користувача: $username</h2>";

        if ($username == 'admin') {
            echo "<p>Прапор: $flag</p>";
            echo "<p>Бот для модерації коментарів - bot.php (при переході за посиланням відправляє цікавий cookie :))</p>";
        }
        ?>
        <p><a class="body-link" href="post.php">Створити новий запис</a></p>
        <p><a class="body-link" href="logout.php">Вийти з облікового запису</a></p>
    </main>
</body>
</html>
```

post.php:

```
<?php
session_start();
?>
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Створення запису</title>
    <link rel="stylesheet" href="/static/styles.css">
</head>
<body>
    <header>
        <div class="header-left">
            <h1><a class='header-link' href="index.php">Forum by Laichuk Artem</a></h1>
        </div>
        <div class="header-right">
            <?php
            if (isset($_SESSION['username'])) {
                echo "<a class='header-link' href='profile.php'>" . $_SESSION['username'] . "</a>";
            } else {
                echo "<a class='header-link' href='login.php'>Увійти/Зареєструватися</a>";
            }
            ?>
        </div>
    </header>
    <main>
        <h1>Створення нового запису</h1>
        <form method="post" action="">
            <label for="title">Заголовок:</label>
            <input type="text" id="title" name="title" required><br>
            <label for="content">Вміст:</label>
            <textarea id="content" name="content" required></textarea><br>
```

```
            <input type="submit" value="Створити запис">
        </form>
        <?php
        if ($_SERVER["REQUEST_METHOD"] == "POST") {
            $title = $_POST['title'];
            $content = $_POST['content'];
            $author_id = $_SESSION['user_id'];

            include 'db.php';

            $sql = "INSERT INTO posts (title, content, author_id) VALUES ('$title', '$content', '$author_id')";
            if ($conn->query($sql) === TRUE) {
                echo "Пост успішно створений! <a href='index.php'>Перейти на головну сторінку</a>";
            } else {
                echo "Error: " . $sql . "<br>" . $conn->error;
            }
            $conn->close();
        }
        ?>
    </main>
</body>
</html>
```

view_post.php:

```
<?php
session_start();
include 'db.php';
?>
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Детальна інформація про запис</title>
    <link rel="stylesheet" href="/static/styles.css">
</head>
<body>
    <header>
        <div class="header-left">
            <h1><a class='header-link' href="index.php">Forum by Laichuk Artem</a></h1>
        </div>
        <div class="header-right">
            <?php
            if (isset($_SESSION['username'])) {
                echo "<a class='header-link' href='profile.php'>" . $_SESSION['username'] . "</a>";
            } else {
                echo "<a class='header-link' href='login.php'>Увійти/Зареєструватися</a>";
            }
            ?>
        </div>
    </header>
    <main>
        <?php
        if (isset($_GET['id'])) {
            $id = $_GET['id'];
            $sql = "SELECT posts.*, users.username AS author_username FROM posts JOIN users ON posts.author_id = users.id WHERE
posts.id = $id";
            $result = $conn->query($sql);
            if ($result->num_rows > 0) {
                $row = $result->fetch_assoc();
                echo "<h2>".$row['title']."</h2>";
                echo "<p>".$row['content']."</p>";
                echo "<p>Автор: ".$row['author_username']."</p>";
            } else {
                echo "Не знайдено жодного запису";
            }
        } else {
            echo "Не передано ID запису.";
        }
        ?>
        <h2>Коментарі:</h2>
```

```
        <?php
        if (isset($_GET['id'])) {
            $post_id = $_GET['id'];
            $sql = "SELECT comments.*, users.username AS commenter_username FROM comments JOIN users ON comments.user_id =
users.id WHERE comments.post_id = $post_id";
            $result = $conn->query($sql);
            while ($row = $result->fetch_assoc()) {
                echo "<p><strong>".$row['commenter_username'].":</strong> ".$row['comment']."</p>";
            }
        }
        ?>
        <form method="post" action="comments.php?post_id=<?php echo $_GET['id']; ?>">
            <label for="comment">Додати коментар:</label>
            <textarea id="comment" name="comment"></textarea><br>
            <input type="submit" value="Залишити коментар">
        </form>
    </main>
</body>
</html>
```

comments.php:

```
<?php
session_start();
include 'db.php';

if ($_SERVER["REQUEST_METHOD"] == "POST" && isset($_GET['post_id'])) {
    $comment = $_POST['comment'];
    $post_id = $_GET['post_id'];
    $user_id = $_SESSION['user_id'];

    $sql = "INSERT INTO comments (post_id, user_id, comment) VALUES ('$post_id', '$user_id', '$comment')";
    if ($conn->query($sql) === TRUE) {
        // Перенаправлення у випадку успішного додавання коментаря
        header("Location: view_post.php?id=$post_id");
        exit();
    } else {
        echo "Error: " . $sql . "<br>" . $conn->error;
    }
}
?>

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Коментарі</title>
    <link rel="stylesheet" href="/static/styles.css">
</head>
<body>
    <header>
        <div class="header-left">
            <h1><a class='header-link' href="index.php">Forum by Laichuk Artem</a></h1>
        </div>
        <div class="header-right">
            <?php
            if (isset($_SESSION['username'])) {
                echo "<a class='header-link' href='profile.php'>" . $_SESSION['username'] . "</a>";
            } else {
                echo "<a class='header-link' href='login.php'>Увійти/Зареєструватися</a>";
            }
            ?>
        </div>
    </header>
    <main>
        <h1>Коментарі:</h1>
        <form method="post" action="">
            <label for="comment">Коментар:</label>
            <textarea id="comment" name="comment" required></textarea><br>
            <input type="submit" value="Submit">
        </form>
```

```
        <?php
        if (isset($_GET['post_id'])) {
            $post_id = $_GET['post_id'];
            $sql = "SELECT comments.comment, users.username FROM comments JOIN users ON comments.user_id = users.id WHERE
post_id = '$post_id'";
            $result = $conn->query($sql);

            if ($result->num_rows > 0) {
                while ($row = $result->fetch_assoc()) {
                    echo "<div class='comment'>";
                    echo "<p>".$row['username'].": ".$row['comment']."</p>";
                    echo "</div>";
                }
            } else {
                echo "Коментарів не знайдено.";
            }
        } else {
            echo "Не передано ID запису.";
        }
        ?>
    </main>
</body>
</html>
```

Скрипт створення бази даних:

```
CREATE DATABASE sql_laichuk;
USE sql_laichuk;

CREATE TABLE users (
    id INT AUTO_INCREMENT PRIMARY KEY,
    username VARCHAR(50) NOT NULL,
    password VARCHAR(50) NOT NULL
);
CREATE TABLE posts (
    id INT AUTO_INCREMENT PRIMARY KEY,
    title VARCHAR(100) NOT NULL,
    content TEXT NOT NULL,
    author_id INT NOT NULL,
    FOREIGN KEY (author_id) REFERENCES users(id)
);
CREATE TABLE comments (
    id INT AUTO_INCREMENT PRIMARY KEY,
    post_id INT NOT NULL,
    user_id INT NOT NULL,
    comment TEXT NOT NULL,
    FOREIGN KEY (post_id) REFERENCES posts(id),
    FOREIGN KEY (user_id) REFERENCES users(id)
);
CREATE TABLE flags (
    id INT AUTO_INCREMENT PRIMARY KEY,
    name VARCHAR(100) NOT NULL,
    data VARCHAR(100) NOT NULL
);
```