Міністерство освіти і науки України
Харківський національний університет ім. В. Н. Каразіна
Факультет комп'ютерних наук
Спеціальність 125 «Кібербезпека»
Освітня програма «Кібербезпека»

"Допущено до захисту"
В.о. завідувача кафедри БІСТ, доцент
Мелкозьорова О.М._____
"_____" червня 2024р.

Пояснювальна записка

до кваліфікаційної роботи бакалавра
на тему: «Розробка методики тестування для виявлення та усунення недоліків веб-сайтів у контексті кібербезпеки»

оцінка « » Керівник ст. викладач Шеханін К. Ю.
Голова ЕК
Лемешко О. В._____ Рецензент PhD Родінко М. Ю.
Виконавець: студентка групи КБ-41

_____ Шадріна С. К.

Харків – 2024

РЕФЕРАТ

Пояснювальна записка до дипломного проекту містить 56 сторінок, 39 рисунків, 3 таблиці, 3 діаграми, 3 додатки та 42 посилання на джерела.

Метою даної дипломної роботи є розробка методики тестування для виявлення вразливостей веб-ресурсів, перевірка стійкості та продуктивності системи шляхом навантажувального тестування, тестування доступності веб-сайту для всіх користувачів, аналіз отриманих результатів, а також надання рекомендацій щодо їх усунення.

Предмет включає в себе вивчення різних підходів та інструментів для аналізу та оцінки отриманих результатів у ході тестування веб-сайту, а також розробку рекомендацій для їх усунення і підвищення рівня захищеності, стійкості та доступності.

Об'єктом роботи є проект веб-сайту Харківського національного університету імені В. Н. Каразіна, що піддається тестуванню розробленою методикою.

У результаті тестування виявлено основні вразливості веб-сайту, що можуть бути використані зловмисниками, проведено навантажувальне тестування та виявлено слабкі місця, проведено тестування доступності, де знайдено помилки, що заважають сприймати вміст сайту без перешкод всім користувачам, а також надано рекомендації для усунення виявлених недоліків.

Результати тестування можуть бути використанні для кращого розуміння та виправлення недоліків ресурсу.

Ключові слова: APACHE JMETER, HOSTEDSCAN SECURITY, OWASP, WAVE, АТАКА, АНАЛІЗ, КІБЕРБЕЗПЕКА, НАВАНТАЖУВАЛЬНЕ ТЕСТУВАННЯ, РОЗРОБКА МЕТОДИКИ, СКАНЕРИ ВРАЗЛИВОСТЕЙ, ТЕСТУВАННЯ ДОСТУПНОСТІ.

ABSTRACT

The explanatory note to the diploma project contains 56 pages, 39 figures, 3 tables, 3 diagrams, 3 appendices and 42 references.

The purpose of this thesis is to develop a testing methodology for identifying web resource vulnerabilities, checking the stability and performance of the system through load testing, testing the availability of the website for all users, analysing the results obtained, and providing recommendations for their elimination.

The subject includes the study of various approaches and tools for analysing and evaluating the results obtained during website testing, as well as developing recommendations for their elimination and increasing the level of security, resilience and availability.

The object of the work is the project of the website of V. N. Karazin Kharkiv National University, which is subjected to testing by the developed methodology.

As a result of the testing, the main vulnerabilities of the website that can be exploited by intruders were identified, load testing was carried out and weaknesses were identified, accessibility testing was carried out, where errors were found that prevent all users from perceiving the content of the site without interference, and recommendations were made to eliminate the identified shortcomings.

The test results can be used to better understand and correct the shortcomings of the resource.

Keywords: APACHE JMETER, HOSTEDSCAN SECURITY, OWASP, WAVE, ATTACK, ANALYSIS, CYBERSECURITY, LOAD TESTING, METHODOLOGY DEVELOPMENT, VULNERABILITY SCANNERS, ACCESSIBILITY TESTING.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

API – Application Programming Interface

CI/CD – Continuous Integration/Continuous Delivery

CORS – Cross Origin Resource Sharing

CSRF – Cross-Site Request Forgery

DAST – Dynamic Application Security Testing

DDoS – Distributed Denial of Service

DoS – Denial of Service

EL – Expression Language

HTML – HyperText Markup Language

HTTP – HyperText Transfer Protocol

HTTPS – HyperText Transfer Protocol Secure

IDS/IPS – Intrusion Detection/Prevention System

ІТС – інформаційно-телекомунікаційні системи

LDAP – Lightweight Directory Access Protocol

MAC – Media Access Control

Nmap TCP UDP – Network Mapper Transmission Control Protocol/ User Datagram Protocol

OGNL – Object Graph Navigation Library

OpenVAS – Open Vulnerability Assessment System

ОС – операційна система

OWASP – Open Web Application Security

OWASP ZAP – Open Web Application Security Project Zed Attack

PDF – Portable Document Format

PHP – Hypertext Preprocessor

ПЗ – програмне забезпечення

SAST – Static Application Security Testing

SQL – Structured Query Language

SSRF – Server Side Request Forgery

SSL – Secure Sockets Layer

SSLyze – SSL/TLS Scanner

СЗІ – системи захисту інформації

TLS – Transport Layer Security

URL – Uniform Resource Locator

VPN – Virtual Private Network

WAF – Web Application Firewall

WAVE – Web Accessibility Evaluation Tool

WCAG – Web Content Accessibility Guidelines

XSS – Cross-Site Scripting

ЄС – Європейський Союз

ЗУ – Закон України

ВСТУП

У сучасному світі використання інтернету стало невід'ємною частиною нашого життя. Різноманітні веб-сайти щодня застосовуються для широкого спектру завдань: від соціальних мереж і електронної комерції до надання державних послуг та освітніх платформ. Проте, зі зростанням залежності від цифрових технологій зростає і ризик кібератак, що вимагає ефективних методів захисту веб-ресурсів від потенційних загроз.

Кожен веб-сайт має свої уразливості. Це легко пояснюється тим, що на етапі розробки людина може зробити помилки, через людський фактор, неуважність, роботу в команді або ще через багато різним причин.

Знаючи основні види вразливостей, які можуть бути допущені, можна легко отримати доступ до будь-якого веб-ресурсу. Цим і користуються зловмисники. Наприклад, за допомогою атак типу XSS хакер може перенаправити запити користувачів на шкідливі веб-сторінки, за допомогою SQL-ін’єкцій – витягувати з баз даних сайту різну конфіденційну інформацію, а за допомогою DDoS-атаки перевантажити системи, після чого вона не може функціонувати належним чином. І таких атак існує ще величезна кількість, з кожним днем вони розвиваються та стають більш складними для відбиття.

Кожен рік кількість атак збільшується, а зловмисники розробляють все більше нових методів атак так, щоб залишитись непоміченими. Протягом 2023 року кількість кіберінцидентів зросла на 62,5% у порівнянні з 2022 роком. Такі дослідження проводять українські аналітики Держспецзв’язку. А до 2025 року щорічні збитки від кібератак зростуть до 10,5 трлн доларів. Відповідно до зібраних статистичних даних, найбільше кіберінцидентів стосувалися шкідливого програмного коду, а саме 28%, збору інформації зловмисниками – 18% та шахрайства – 6% [17].

З цього і випливає висновок, що актуальність теми дослідження полягає у постійному розвитку та ускладненні методів кібератак, що вимагає відповідного розвитку методів захисту та тестування вразливостей. З огляду на широкий

спектр можливих загроз, питання оцінки захищеності веб-сайтів є вкрай важливим для забезпечення конфіденційності, цілісності та доступності інформації у наш час.

Для того, щоб виявити і усунути різні види веб-вразливостей фахівці проводять тестування веб-сайту. У даній дипломній роботі буде розроблена методика тестування веб-ресурсів для виявлення різних видів недоліків, які можуть призвести до серйозних атак та аналіз виявлених ризиків. Також будуть запропоновані рекомендації щодо їх усунення задля безпеки.

1 ТЕОРЕТИЧНІ ОСНОВИ ЗАГРОЗ ТА ЗАХИСТУ ВЕБ-САЙТІВ

1.1 Огляд основних понять кібербезпеки

Сьогодні кібербезпека відіграє велику роль у захисті інформаційних систем та конфіденційних даних від різного виду кібератак. Але що таке саме поняття кібезбезпеки?

Кібербезпека – це комплекс процесів, практичних порад і технологічних рішень, які допомагають захищати важливі системи й мережу від кібератак [1].

Кожного дня злочинці розробляють нові методи атак для того, щоб отримувати доступ до конфіденційних даних користувачів. Кібербезпека у свою чергу впроваджує інноваційні методики захисту. Це надає користувачам змогу працювати, спілкуватись, робити покупки онлайн безпечно.

Розглянемо ключові поняття кібербезпеки, які знадобляться нам у процесі дослідження.

Вразливість – це слабкість у системі, яку може використати зловмисник для втручання в нормальну роботу системи.

Загроза – це потенційна подія, яка може завдати шкоди системам або організації, використовуючи вразливість.

Кібератака – це спрямована дія у кіберпросторі з утручанням у роботу ІТС, що використовує вразливості системи для досягнення зловмисних цілей, з метою порушення конфіденційності, цілісності, доступності інформації.

Шифрування – це один із основних методів захисту даних, процес перетворення інформації у форму, що може бути прочитана лише особою, яка має ключ шифрування.

Мережевий брандмауер – це безпековий пристрій, який контролює вхідний і вихідний мережевий трафік на основі визначених правил безпеки.

Керування доступом – це процес, який гарантує, що доступ до конкретних ресурсів надається лише тим, хто має на це право.

Саме розуміння таких ключових понять кібербезпеки є вкрай важливим аспектом для розробки ефективних стратегій захисту інформації.

1.2 Види загроз кібербезпеці веб-сайтів

Сьогодні безпека веб-сайтів – це одне з найбільш важливих питань в контексті інформаційної безпеки. Як правило, більшість ресурсів, які знаходяться у вільному доступі інтернеті, мають різного роду вразливості і майже кожен день піддаються кібератакам.

Розглянемо основні типи загроз для інформаційної безпеки веб-сайтів:

- Загрози конфіденційності або несанкціонований доступ до даних.

Основними шляхами порушення конфіденційності можуть бути втрата контролю над СЗІ або різні канали витоку інформації.

- Загрози цілісності або несанкціонована модифікація, знищення даних.

Для даного виду загроз основним шляхом порушення також є канали витоку інформації. Але так, як основною відмінністю від загрози конфіденційності є саме модифікація даних, розглянемо поняття «канал дії на цілісність». Це спосіб, завдяки якому зловмисник може змінювати дані, видаляти їх або вводити шкідливий код. Прикладом такого каналу може слугувати використання програми «троянський кінь».

- Загрози доступності або обмеження, блокування доступу до даних.

Даний вид загрози може виникати внаслідок порушень безпеки. Наприклад, порушення працездатності елементів автоматизованих систем, недосконаліть конструкції, зловмисні дії людей що спрямовані на порушення безпеки інформації тощо.

Основною загрозою безпеці веб-сайту є хакерська атака, визначення якої було розібрано вище, у пункті 1.1. Зазвичай використовуються декілька поверхових або найбільш очевидних вразливостей.

На сьогоднішній день існує величезна кількість різних видів атак, які використовують зловмисники з метою контролю, зміни в роботі, модифікації або знищення інформації для свої корисних цілей. Це може бути привернення уваги для чогось, розповсюдження шкідливого ПЗ, вимагання грошової винагороди тощо.

1.3 Огляд вразливостей веб-ресурсів

В 21 столітті майже кожна людина має вільний доступ до Інтернету, публікуючи інформацію у вільний доступ. Задля отримання своєї вигоди зловмисники постійно атакують веб-ресурси різної значимості: від приватної сторінки школяра у соціальних мережах до атак на крупні світові компанії.

Вразливості у веб-сайтах давно становили небезпеку для користувачів. Отже, розглянемо які саме вразливості і слабкі місця може містити у собі веб-ресурс, що може призвести до успішних кібератак. Даний топ вразливостей на 2021 рік дослідила організація OWASP [3]:

a) Порушення контролю доступу.

Контроль доступу забезпечує дотримання політики таким чином, щоб користувачі не могли діяти поза межами призначених дозволів. Збої часто призводять до несанкціонованого оголошення інформації, її модифікації або видалення всіх даних.

b) Криптографічні збої.

Ця вразливість є головною причиною витоку конфіденційної інформації (наприклад, паролі, номери та пін-коди від банківських карток, особиста інформація тощо) під час передавання та зберігання. Ці дані вимагають додаткового захисту і підпадають під дію законів про інформацію у різних країнах. Наприклад, «Загального регламенту захисту даних ЄС», ЗУ «Про захист інформації в інформаційно-телекомунікаційних системах» або інших нормативно-правових актах та стандартах.

c) Ін’єкції.

У даному випадку веб-сайт вразливий до атак якщо дані, які надав користувач, регулярно не перевіряються, не фільтруються та не очищаються; ворожі дані використовуються на пряму або об’єднуються; SQL або команда містить структуру та шкідливі дані в динамічних запитах, командах або збережених процедурах. Одними з найпоширеніших видів ін’єкцій є SQL. Також варто звернути увагу на такі види ін’єкцій як: NoSQL, команди ОС, ORM, LDAP, EL, OGNL тощо.

d) Незахищений дизайн.

Незахищений дизайн — це категорія, що представляє різні слабкі сторони, виражені як «відсутній або неефективний дизайн контролю». Незахищений дизайн неможливо виправити ідеальною реалізацією, оскільки необхідні засоби контролю безпеки не створювалися для захисту від конкретних атак. Однією з причин, які сприяють незахищеному дизайну, є відсутність профілювання бізнес-ризиків, що притаманне програмному забезпеченню чи системі, що розробляється. Як висновок – нездатність визначити, який рівень безпеки дизайну потрібен.

e) Неправильна конфігурація безпеки.

Дана вразливість спостерігається, якщо відсутнє потрібне посилення безпеки в будь-якій частині стеку програм або ж дозволи для хмарних служб неправильно налаштовані; встановлено непотрібні функції (такі як непотрібні порти, сторінки, облікові засоби тощо); облікові записи і їхні паролі залишаються активними та не змінюються; нові функції безпеки вимкнено або не налаштовано для систем, що оновились; параметри безпеки на серверах додатків, платформах додатків, бібліотеках, базах даних тощо не мають безпечних значень.

f) Вразливі та застарілі компоненти.

Веб-сайт може бути вразливим у наступник випадках: якщо ви не знаєте версії всіх компонентів, які використовуєте; ПЗ є вразливим, не підтримується або його версія вже застарілі; сканування на наявність вразливостей не проводиться регулярно; відсутнє оновлення базової платформи, фреймворків та залежностей вчасно з урахуванням ризиків; розробники ПЗ не перевіряють сумісність оновлених або виправлених бібліотек.

g) Помилки ідентифікації та автентифікації.

Підтвердження особи користувача, автентифікація та керування сеансами мають дуже важливу роль для захисту від атак, які пов’язані з автентифікацією. Такі недоліки можуть бути виявлені у разі якщо: дозволено автоматизовані атаки, (наприклад, перекидання облікових даних, коли зловмисник має список

дійсних імен користувачів і паролів); дозволена атака brute force; використання стандартний або слабких паролів, неефективні процедурі відновлення облікових записів та паролів; відсутня багатофакторної автентифікації; розкрито ідентифікатор сеансу в URL-адресі; після успішного входу ідентифікатор сеансу використовується повторно.

h) Порушення цілісності ПЗ та даних.

Незахищеність коду та інфраструктури, які не вберігаються від втручань у цілісність, стосується ситуацій, коли ПЗ та дані не захищені від атак, які порушують їх цілісність, належним чином. Прикладом може слугувати ситуація, коли програма використовує плагіни, бібліотеки або модулі з ненадійних джерел. Незахищений конвеєр CI/CD може призвести до несанкціонованого доступу, зловмисного коду або компрометації системи. Так потенційно зловмисники можуть завантажити власні оновлення для розповсюдження та запуску на всіх інсталяціях. Іншим прикладом є те, що об’єкти або дані кодуються або серіалізуються в структуру, яку зловмисник може побачити та змінити, вразливу до незахищеної десеріалізації.

i) Помилки в журналі та моніторингу безпеки.

Ця категорія має на меті допомогти виявляти, розголошувати та реагувати на активні порушення. Без реєстрації в журналі та моніторингу порушення неможливо виявити. Недостатнє ведення журналу, виявлення та моніторингу відбувається якщо: події, які підлягають аудиту не реєструються, попередження та помилки не створюють жодних незрозумілих повідомлень журналу; журнали програм і API не перевіряються на наявність підозрілої активності; журнали зберігаються лише локально; тестування на проникнення та сканування інструментами динамічного тестування безпеки додатків не викликають попереджень; програма не може виявляти або сповіщати про активні атаки в режимі реального часу.

j) Підробка запитів на стороні сервера.

Помилки SSRF виникають, коли веб-програма отримує віддалений ресурс без перевірки наданої користувачем URL-адреси. Це дозволяє зловмиснику

зробити так, щоб програма надіслала створений запит до несподіваного адресата, навіть якщо він захищений брандмауером, VPN або іншим типом контролю доступу до мережі.

Оскільки новітні веб-ресурси надають кінцевим користувачам зручні функції, то отримання URL-адреси стає звичайним сценарієм. Як наслідок, інциденти з підробкою запитів на стороні сервера зростають. Крім того, серйозність цієї помилки стає вищою через хмарні сервіси та складність архітектур.

1.4 Види атак, що використовуються зловмисниками

В епоху цифровізації, коли всі аспекти нашого життя так чи інакше пов’язані з комп’ютерними технологіями, кібербезпека стала невід’ємною частиною захисту інформації, методи якої постійно розвиваються. Але так само розвиваються і методи атак, які використовуються хакери для проведення атак. Розуміння різних видів атак, які можуть бути використані проти інформаційних систем, є критично важливим для розробки ефективних стратегій захисту.

Зловмисники використовують різноманітні техніки і тактики для досягнення своїх цілей. Розглянувши найпоширеніші види вразливостей, які використовують хакери для кібератак, детально розглянемо найпоширеніші види атак, які використовують кіберзловмисники.

1.4.1 SQL-ін’єкції

SQL-ін’єкції – одна з найпопулярніших методик злому веб-сайтів та програм, що базуються на роботі з базами даних. Веб-програми, які не розроблені ретельно (з точки зору безпеки), схильні до атак SQL-ін’єкцій. Згідно з деякими дослідженнями, які проводили Positive Technologies [5], майже 70% веб-сайтів піддаються кібератакам, де одне з перших місць посідають саме SQL-ін’єкції.

SQL-ін’єкцій — це тип ін’єкцій коду у програмі, під час якого зловмисники додають шкідливий код до коду користувача, щоб отримати неавторизований і необмежений доступ. Атака на основі SQL-ін’єкції проводиться шляхом

надсилання запитів до бази даних на основі даних, які вводить користувач. Код зловмисників передається таким чином, що він стає запитом. SQL-ін'єкції небезпечні, адже вони відкривають хакерам можливість робити все, що завгодно. Схема роботи цієї атаки зображена на рис. 1.1.

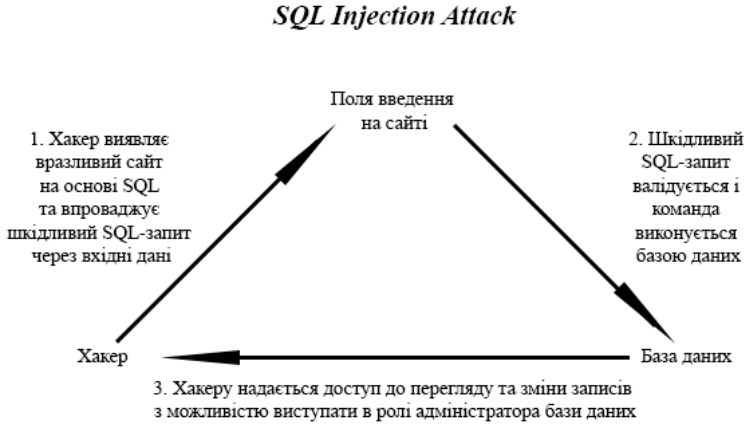


Рисунок 1.1 – Схема атаки SQL [7]

Якщо SQL-ін'єкція виконана успішно, то неавторизовані особи з легкістю отримають доступ до конфіденційних даних, а також зможуть їх читати, оновлювати або змінювати та видаляти записи з бази даних.

1.4.2 Cross-Site Scripting (XSS або міжсайтовий скриптинг)

Наступною популярною атакою, яку я хочу розглянути буде міжсайтовий скрптинг або XSS.

XSS — це вразливість, яка дозволяє зловмиснику виконувати шкідливі коди на веб-сайті. Це може вплинути на веб-сайт або викрасти конфіденційні дані з веб-сайту або заволодіти обліковим записом користувача. Ці атаки можуть призвести до різних наслідків, включаючи крадіжку даних, пошкодження веб-сайту та поширення шкідливого програмного забезпечення. Оскільки популярність веб-додатків продовжує зростати, важливість розуміння методів атак XSS і стратегій запобігання стає все більш критичною.

Цей вид атаки може бути можливим, коли створені веб-сторінки відображають вхідні дані, які не перевірені належним чином. Це дозволяє зловмиснику, вставити свій зловмисний JavaScript код у згенеровану сторінку та виконати сценарій на комп'ютері будь-якого користувача, який переглядає цей сайт.

Будь-який веб-сайт може бути вразливим до цього типу атаки. Вона може бути здійснена через різні напрямки. Наприклад через поля введення, файли cookie та URL-адреси, і може вплинути на всіх користувачів, які відвідують уражену веб-сторінку. Таким чином, XSS став однією з найважливіших проблем безпеки для веб-розробників і користувачів. Схема роботи XSS атаки зображена на рис 1.2.

Рисунок 1.2 – Схема атаки Cross-Site Scripting [8]

1.4.3 Cross-Site Request Forgery (CSRF)

CSRF – це вид атаки, де хакер змушує користувачів виконувати небажані дії на веб-сайті, де користувач вже автентифікований. Запустивши успішну CSRF-атаку проти автентифікованого користувача, зловмисник може ініціювати довільні запити HTTP з використанням облікових даних користувача до вразливого веб-додаток. Успішна атака CSRF ефективно обходить основний механізм автентифікації, що може поставити під загрозу дані кінцевого користувача.

Якщо розібрати простими словами, то цей тип атак спрямований на імітування запиту користувача до стороннього веб-сайту. Ця вразливість досить широко поширена через те, що багато веб-застосунків не чітко визначають – чи дійсно запит був сформований справжнім користувачем.

Атаки CSRF зазвичай такі ж сильні, як і користувач, тобто будь-яка дія, яку може виконати користувач, також може бути виконана зловмисником за допомогою атаки CSRF. Отже, чим більше можливостей сайту надає користувачеві, тим серйознішими є можливі CSRF-атаки. Наприклад, якщо

обліковий запис жертви має права адміністратора, це може скомпрометувати всю веб-програму. Схема роботи CSRF зображена на рис. 1.3.

Рисунок 1.3 – Схема атаки Cross-Site Request Forgery [11]

1.4.4 Denial of Service (Dos)

DoS-атака – це тип кібератаки, під час якої зловмисник виводить з ладу або перевантажує систему, сервер чи мережеву інфраструктуру так, щоб легітимні користувачі не могли до них отримати доступ.

Головна мета атаки Dos – це порушення нормального обслуговування об’єкта шляхом перевантаження об’єкта або навколишньої інфраструктури потоком Інтернет-трафіку. Найочевиднішим показником DoS-атаки є те, що сайт або служба раптово стають повільними або недоступними. Будь-яка галузь чутлива до атак DoS/DDoS через використання Інтернету.

Відмова в обслуговуванні є фундаментальною проблемою інформаційної безпеки, оскільки вона перешкоджає доступності інформації в потрібний час. Схема роботи цієї атаки зображена на рис 1.4.

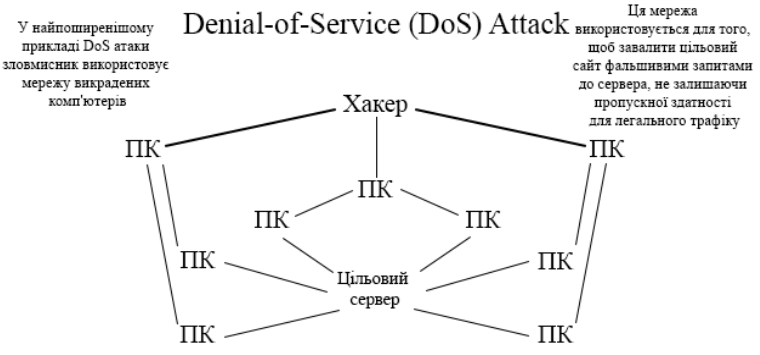


Рисунок 1.4 – Схема атаки Denial of Service [12]

1.4.5 Distributed Denial of Service (DDoS)

Атака типу DDoS є більш удосконаленою версією DoS-атаки. Головною їх відмінністю є те, що це розподілена атака, а, отже, зловмисник використовує декілька джерел для перевантаження мережі, яку він атакує. Але на меті у хакера те саме, що і під час DoS-атаки – перевантажити систему так, щоб у користувачів не було доступу до потрібного веб-сайту.

Порівняно зі звичайними DoS-атаками, які можна вирішити за допомогою кращого захисту систем обслуговування або заборони несанкціонованого віддаленого чи локального доступу, DDoS-атаки є складнішими і їм важче запобігти. Ці атакаи вважаються більш руйнівними через їх здатність залучати велику кількість компрометованих систем для генерації величезного обсягу трафіку, який може перевантажити навіть добре захищені ресурси.

Останніми роками DDoS-атаки зросли за частотою та серйозністю через той факт, що вразливість комп’ютерів швидко зростає. Тому цей тип атаки є одним з найскладніших, з якими може зіткнутися організація, оскільки вона об'єднує ресурси з багатьох різних джерел, ускладнюючи процес виявлення та блокування атаки. Схема роботи DDoS-атаки зображена на рис 1.5.

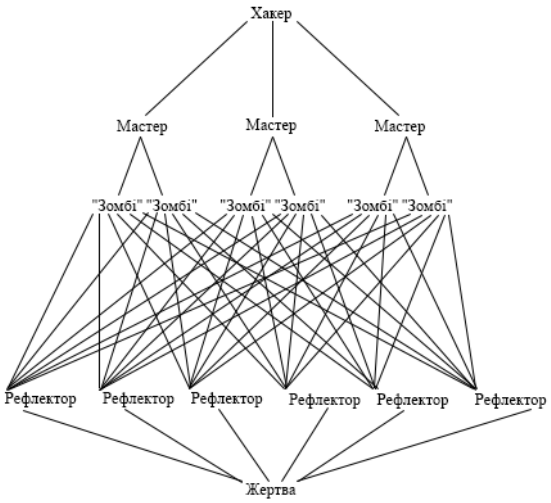


Рисунок 1.5 – Схема атаки Distributed Denial of Service [13]

1.5 Методи та інструменти захисту веб-сайтів від кібератак

З появою проблем, які пов’язані з витоком даних та різноманітними кібератаками, забезпечення безпеки веб-сайтів стає дедалі актуальнішою

задачею для фахівців в цій сфері. З цього випливає, що критично важливим аспектом у захисті конфіденційних даних є впровадження перевірених методів та інструментів безпеки.

Першим методом захисту, який я хочу розглянути, є регулярне оновлення програмного забезпечення. Саме своєчасне оновлення ПЗ допомагає виявити та усунути відомі або вже застарілі вразливості веб-застосунку. Адже зловмисники часто використовують вразливості саме у застарілому ПЗ, щоб отримати несанкціонований доступ до вашого ресурсу.

Другим методом можна виділити використання протоколу HTTPS замість стандартного HTTP. Цей протокол забезпечує шифрування даних, що передаються між сервером та користувачем, використовуючи SSL та TLS сертифікати для забезпечення конфіденційності та цілісності даних під час їх передачі через Інтернет.

Встановлюйте сильні паролі, які складно буде зламати зловмисникам, а також використовуйте багатофакторну автентифікацію. Ці заходи забезпечують додаткові рівні безпеки, ускладнюючи зловмисникам можливість отримати доступ до ваших даних.

Останнім з методів, який я хочу розглянути, є резервне копіювання даних. Це необхідно для того, щоб ваші важливі дані збереглись навіть у разі проведення зловмисником успішної атаки.

Далі розглянемо деякі з інструментів захисту веб-сайтів. Першим з таких інструментів є використання мережевих захисних механізмів. Цей інструмент є одним з найпотужніших стратегій захисту від кібератак, адже він допомагає ідентифікувати, блокувати або пом'якшувати атаки, забезпечуючи безпеку даних та зберігаючи доступність ресурсів. Найпоширеніші види цього інструменту:

- Фаєрволи або мережеві екрани допомагають запобігати несанкціонованому доступу, контролють вхідний та вихідний трафік мережі, блокують підозрілу активність. Прикладами цього інструменту є IPTables, Windows Firewall тощо;

- Брандмауер веб-додатків WAF фільтрує вхідний HTTP-трафік, блокуючи підозрілі запити та спроби використання вразливостей. Він є ефективним при захисті від атак типу SQL-ін'єкцій, XSS та CSRF, розглянутих вище. Прикладами є ModSecurity, Imperva, Cloudflare WAF;
- Системи виявлення та запобігання вторгненням IDS/IPS, де IDS перевіряє мережевий трафік на предмет відомих атак, а IPS автоматично вживає заходи для блокування зловмисного трафіку. Прикладами є Snort, Suricata.

Дуже важливим інструментом для захисту веб-сайтів є тестування. Це може бути тестування на проникнення, статичний та динамічний аналіз коду, використання автоматизованих інструментів тощо. Це допомагає виявити ризики та вразливості веб-сайту, а також вирішити їх ще на етапі розробки. Прикладом може бути безкоштовний інструмент для тестування OWASP ZAP, який автоматично сканує веб-додатки на наявність таких вразливостей як SQL-ін'єкції, XSS, CSRF, дає можливість ручного тестування додатка та надає рекомендації щодо усунення виявлених проблем Але найбільшим забезпеченням безпеки будь-якого веб-сайту буде вважатись поєднання всіх методів в комплексний підхід для тестування.

2 ОГЛЯД МЕТОДІВ ТЕСТУВАННЯ

2.1 Огляд методів тестування безпеки веб-сайтів

Тестування безпеки веб-сайтів – це дуже важлива частина розробки веб-додатків. Вона дозволяє знайти та усунути вразливості, які були допущені та можуть бути використані зловмисниками у їх цілях. Розглянемо деякі з основних поширених методів тестування, щоб зрозуміти як вони працюють.

- Статичний аналіз коду (SAST)

Це процес автоматичного сканування вихідного коду на наявність вразливостей у ньому. Цей метод аналізує синтаксис і структуру коду, щоб виявити потенційні вразливості, такі як помилки в синтаксисі, недостатня валідація даних, недостатня авторизація та інші проблеми, які можуть бути виявлені на ранніх етапах розробки. Цей вид тестування сканує вихідний код методом «білого ящика». Тобто тестувальник має доступ до вихідного коду програми і використовує цю інформацію для планування та виконання різних тестів без фактичного виконання програми.

SAST на основі аналізу ще скомпільованого коду допомагає виявити та усунути вразливості до того, як програма буде передана до подальшого тестування, визначає критичні вразливості типу SQL та XSS, синтаксичні помилки, а також дозволяє покращити якість коду, відповідно і усього проекту загалом.

Ручний аудит і перевірка коду також можуть підпадати під категорію статичного аналізу, але вони залишаються неефективними, доки ця діяльність не буде автоматизована, що робить її швидшою та надійнішою [16]. Тому статичний аналіз часто виконується за допомогою спеціального ПЗ, яке дозволяє швидко проаналізувати великі обсяги коду і виявити потенційні ризики, надаючи тестувальникам детальні звіти про помилки та вразливості.

- Динамічний аналіз коду (DAST)

На відміну від статичного аналізу коду, процес динамічного аналізу виконується на активному коді в реальному часі. Тобто DAST в свою чергу – це

процес тестування веб-додатку під час його виконання. Таким чином цей вид тестування дозволяє виявити вразливості, які можуть бути виявленими лише під час виконання програми.

Даний метод аналізує веб-додаток під час його виконання, імітуючи зовнішні атаки, які націлені на використання поширених вразливостей, наприклад, SQL-ін'єкції, XSS-уразливості та інші. Головна задача динамічного тестування – виявити незаплановані ризики раніше, ніж це зможуть зробити кіберзловмисники.

Методи моніторингу під час виконання більш доцільні для запобігання атакам під час виконання, а не для їх виявлення. Його не можна використовувати для виявлення розташування вразливостей, але це може допомогти запобігти використанню вразливості під час виконання програми [16].

- Ручне тестування

Ручне тестування – це тип тестування, який виконується вручну безпосередньо фахівцем, тобто тестувальником. Даний метод передбачає виконання тестових кейсів без використання інструментів для автоматизованого тестування. Мануальне тестування використовується, щоб впевнитись, що ПЗ відповідає всім стандартам якості [18].

У ході роботи тестувальник відіграє роль звичайного користувача програми, що дає змогу виявити вразливості і помилки, які можуть бути невидимими для автоматизованого тестування. Це може включати в себе тести на візуальне сприйняття, тести відмови користувача тощо.

Недоліками ручного тестування є те, що воно вимагає більше часу та є менш ефективним у порівнянні з автоматизованим.

- Автоматизоване тестування

Автоматизоване тестування – це тип тестування ПЗ, який використовує програмні інструменти для автоматичного виконання тестових кейсів. Воно передбачає використання спеціального ПЗ для виконання тестових завдань замість ручної роботи. Автоматизоване тестування використовується для

підвищення ефективності та точності тестування програмного забезпечення за рахунок зменшення кількості необхідних ручних зусиль [17].

Процес автоматизації включає планування, проектування та розробку скриптів для автоматизації, спрямованої на зменшення кількості необхідних ручних тестів, а не на повне усунення ручного тестування.

Автоматизоване тестування має ключове значення для підвищення ефективності, тестового покриття та швидкості процесів тестування ПЗ.

2.2 Порівняльний аналіз методів

Для того, щоб наочно побачити переваги та недоліки розглянутих методів тестування, складемо порівняльну таблицю (таб. 1) [19, 20].

Таблиця 2.1 – Порівняльний аналіз методів тестування

	Статичний метод	Динамічний метод	Ручне тестування	Автоматизоване тестування
Переваги	Виявлення помилок на ранній стадії розробки у структурі коду	Виявлення вразливостей під час реальної роботи програми	Дозволяє досліджувати різні частини ПЗ та виконувати різні сценарії	Автоматизоване та швидке виконання тестів
	Покращення якості вихідного коду	Перевірка функціональності програми	Інтуїтивне виявлення дефектів, яке може пропустити програма	Більш надійні тести, адже відсутні помилки через людський фактор
	Аналіз коду, документації без запуску програми	Автоматизація більшості процесів тестування	Швидка зміна тестових сценаріїв	Ефективне для повторюваних тестів

Продовження таб. 2.1.

Недоліки	Не гарантує повного виявлення помилок	Вимагає запуску програми для виявлення помилок	Помилки через людський фактор	Постійне оновлення скриптів
	Обмежені можливості для автоматизації	Може не виявляти помилки, які не викликаються під час запуску	Обмежене у виявленні проблем, що пов’язані з великим обсягом даних	Початкові витрати на розробку тестових скриптів
	Може займати багато часу через великий обсяг коду	Тестування проходить на більш пізніх етапах розробки	Обмежена масштабованість через великий обсяг	Деякі варіанти могли бути проігноровані та не враховані при розробці тестових сценаріїв

З цієї таблиці (таб. 1) видно, що статичне і динамічне тестування є доповненням одне одного. Але різниця полягає в тому, що статичне тестування більш ефективне для виявлення дефектів на ранніх стадіях розробки, тоді як динамічне тестування краще підходить для перевірки фактичної поведінки програми. В ідеалі обидва методи слід поєднувати, щоб забезпечити комплексне тестування ПЗ.

Далі розглянемо ручне та автоматизоване тестування. Перший метод тестування пропонує гнучкість і використовує людські навички, але він є більш затратним і повільним. Автоматизоване тестування є швидшим і дешевшим, але має обмежену гнучкість і вимагає початкових інвестицій у скрипти. Для забезпечення комплексного підходу до тестування також слід поєднувати ці два види між собою у роботі.

2.3 Інструменти для автоматизації тестування веб-сайтів

Автоматизація – дуже потужний інструмент для тестування веб-додатків. Він допомагає забезпечити високу ефективність завдяки виконанню регулярних тестів без необхідності ручної перевірки.

Для автоматизації тестування існує ряд інструментів, які допомагають зробити цей процес ефективним. Розглянемо деякі з найпопулярніших методів [21, 22, 23, 24]:

- Selenium

Selenium – це інструмент автоматизації з відкритим вихідним кодом, який широко використовується для тестування веб-додатків. Він підтримує кілька мов програмування (Java, Python, C#, Perl, PHP і JavaScript) та операційних систем і пропонує ряд функцій для тестування веб-додатків.

Це один із найпоширеніших інструментів автоматизованого тестування. Selenium забезпечує функції запису та відтворення тестових сценаріїв, паралельне тестування та інтеграцію з різними фреймворками.

- Apache Jmeter

Apache JMeter – це інструмент для тестування навантаження з відкритим вихідним кодом. Це настільний Java-додаток, призначений для тестування функцій завантаження та вимірювання продуктивності веб-сайтів. Інструмент забезпечує різноманітні графіки та звіти для візуалізації результатів тестування, включаючи графіки продуктивності, діаграми часу відгуку та графіки помилок.

Наразі JMeter широко використовується розробниками, тестувальниками продуктивності та DevOps-інженерами для перевірки навантаження, стресового тестування, тестування стабільності, аналізу продуктивності та функціонального тестування різноманітних додатків та систем.

- Сканери вразливостей

Сканери вразливостей – це спеціалізовані інструменти, призначені для автоматизованого пошуку слабких місць у програмному забезпеченні, мережах або веб-додатках. Він використовується для пошуку потенційних проблем

безпеки в мережах, комп'ютерних операційних системах, базах даних і додатках до того, як їх зможуть використати зловмисники.

Його основними завданнями є оцінка безпеки, виявлення вразливостей і звітування про виконану роботу. Інструмент використовує метод «чорного ящика», адже дані тести не потребують доступу до вихідного коду, а запускають зовнішні атаки для перевірки вразливостей безпеки. Сканер відносять до категорії інструментів динамічного тестування безпеки.

Сканування ПЗ, веб-сайтів та мереж на предмет відомих вразливостей, таких як неправильні конфігурації, застаріле програмне забезпечення, вразливості до ін’єкцій SQL, XSS, CSRF та інші.

- WAVE

Тестування веб-доступності є невід’ємною частиною для простоти використання веб-сайтів та сприйняття його людиною, яка має деякі вади у сприйнятті візуального, звукового або іншого контенту. Для цього слід проводити тестування accessibility або ж тестування доступності.

Для цього виду тестування добре підходить інструмент WAVE, який оцінює доступність веб-сайтів. Він може виявляти потенційні помилки та проблеми, які можуть перешкоджати людям з обмеженими можливостями у повноцінному використанні ресурсу.

3 РОЗРОБКА МЕТОДИКИ ТЕСТУВАННЯ ДЛЯ ВИЯВЛЕННЯ ВРАЗЛИВОСТЕЙ ВЕБ-САЙТІВ

3.1 Визначення вимог до методики

Для забезпечення ефективності процесу моніторингу вразливостей сайту необхідно чітко визначити вимоги до розроблюваної методології. Вимоги повинні охоплювати ключові аспекти, які дозволять систематично виявляти та усувати потенційні загрози безпеці веб-сайту. Розглянемо критерії, які я виділили для розробки своєї методики.

Перший пункт – це всебічність тестування. Розроблена методика має охоплювати комплексний підхід до виявлення вразливостей, що загрожують безпеці ресурсу, виявлення можливих помилок та проблем у доступності та визначення надійності та стійкості системи при великих навантаженнях на неї.

Другий критерій – автоматизація. Методика має передбачати можливість для автоматизації процесів тестування за допомогою спеціалізованих інструментів, що допоможе підвищити ефективність цього процесу, зекономивши час та ресурси.

Третій крок – актуальність. Цей критерій є критично важливим на сьогоднішній день, адже кожен день хакери розробляють все більше атак для несанкціонованого доступу до конфіденційних даних та ресурсів. Для розробки методики необхідно враховувати останні найпопулярніші загрози кібербезпеки та адаптуватись до нових атак та вразливостей, які можуть з’явитись.

Четвертий пункт – гнучкість. Методика також має бути адаптована до різних видів веб-сайтів та вимог, які може мати конкретний веб-ресурс, а також здатна інтегруватись з іншими інструментами та підходами до тестування.

Останній пункт – документування. Врешті-решт, весь процес тестування, а також його результати мають бути задокументовані, включаючи виявлені вразливості та рекомендації щодо їх усунення для подальшого аналізу та удосконалення веб-сайту.

3.2 Опис процесу розробки методики

Розробка ефективної методики тестування веб-сайтів для виявлення вразливостей та недоліків є комплексним процесом, який вимагає ретельного планування та дослідження.

Для комплексного тестування веб-сайту необхідно ознайомитись з найвірогіднішими вразливостями, які сьогодні зустрічаються та які потенційно можуть бути виявлені веб-ресурсі. З такими вразливостями ми вже ознайомились в OWASP Top Ten у пункті 1.3.

Для виявлення розглянутих вразливостей скористаємось сканерами вразливостей. Сканери вразливостей – це програмні або апаратні інструменти, які використовуються для діагностики та моніторингу мережевих комп'ютерів з метою виявлення потенційних вразливостей в системі, які можуть бути використані зловмисниками. Ці інструменти можна використовувати для тестування, оцінки та виправлення вразливостей в різних додатках з метою підвищення безпеки системи.

Ці інструменти важливі для безпеки мережі та даних, оскільки вони виявляють і виправляють вразливості, які можуть бути використані для зовнішніх атак. Вони дозволяють адміністраторам і розробникам додатків забезпечити високий рівень захисту від потенційних загроз.

Існує декілька видів сканерів [25]:

— мережеві сканери (виявляють вразливості у всій мережі організації шляхом аналізу систем, пристроїв, з'єднань, а також портів і служб мережевої інфраструктури, які можуть бути використані);
— сканери на основі хостів (оцінює конфігурації та операційні системи серверів, локальних комп'ютерів і мережевих хостів для виявлення вразливостей і забезпечення видимості налаштувань конфігурації та історії виправлень);
— бездротові сканери (виявляють вразливості в бездротових мережах, забезпечуючи безпеку бездротових з'єднань і пристроїв);

— сканери додатків (тестують мережі, веб-сайти, веб-додатки і мобільні додатки для виявлення відомих помилкових конфігурацій і вразливостей ПЗ);
— сканери баз даних (виявляють слабкі місця в базах даних, скануючи вразливості, такі як помилкові конфігурації безпеки, забезпечуючи цілісність, конфіденційність та доступність баз даних та систем управління базами даних);
— хмарні сканери (сканують хмарні розгортання для моніторингу, управління та підвищення безпеки хмарної інфраструктури організації, зосереджуючись на загальних вразливостях у хмарних налаштуваннях та засобах управління).

Наступним кроком у розробці методики є перевірка веб-сайту на стійкість та надійність під час сильно перевантаження вхідного трафіку на систему. В цьому нам допоможе навантажувальне та стресове тестування.

Навантажувальне тестування – це вид тестування, який полягає в оцінці продуктивності та стійкості програмного забезпечення або системи при збільшенні навантаження, що перевищує очікуване. Тобто це випробування, коли від 2 до 100 користувачів одночасно намагаються отримати доступ до сайту. По суті, воно дозволяє перевірити налаштування серверної системи та визначити максимальний обсяг вхідного трафіку [26].

Основна мета навантажувального тестування – виявити слабкі місця в системі, визначити її межі продуктивності та стабільності, а також оцінити, як система буде працювати в умовах реального використання.

Стресове тестування, в свою чергу – це вид тестування продуктивності, який полягає в дослідженні поведінки програми при несподіваних змінах навантаження, які перевищують розрахований для веб-сайту рівень.

Стрес-тестування ПЗ є важливою частиною процесу тестування і призначене для виявлення вразливостей, слабких місць і потенційних помилок, які можуть виникнути, коли система піддається високому навантаженню або несприятливим умовам [27].

Завершуючим етапом розробки методики є тестування доступності веб-сайту, оскільки цей етап є вкрай важливим для забезпечення рівного доступу до інформаціїї для всіх користувачів.

Тестування доступності – це процес тестування ПЗ або веб-сайтів на доступність для людей з обмеженими можливостями, такими як люди з порушеннями зору, слуху або руховими обмеженнями.

Цей вид тестування проводиться згідно з вимогами та стандартами, які широко використовуються у всьому світі. Найвідомішим стандартом є WCAG. Розглянемо основні його принципи [28]:

— Сприйнятність (Perceivable). Це означає, що інформація та компоненти інтерфейсу користувача повинні бути представлені користувачам у формах, які вони можуть сприймати;
— Керованість (Operable). Компоненти інтерфейсу користувача та навігація повинні бути керованими;
— Зрозумілість (Understandable). Інформація та управління інтерфейсом користувача повинні бути зрозумілими;
— Надійність (Robust). Контент повинен бути достатньо надійним, щоб його можна було інтерпретувати широким колом користувачів.

WCAG пропонує три рівні доступності: A, AA, AAA, де A – мінімальний рівень, AA – середній та AAA – найвищий рівень доступності.

3.3 Переваги та недоліки розробленої методики

Для більш наглядного огляду розробленої методики слід визначити її основні переваги та недоліки. Розглянемо їх у таб. 3.1.

Таблиця 3.1 – Переваги та недоліки розробленої методики

Переваги	Недоліки
Методика дозволяє комплексно підійти до тестування, виявлення різних типів вразливостей і перевірку стабільності та доступності	Складність підходу та використання різних інструментів може ускладнити реалізацію, особливо в невеликих проектах

Продовження таб. 3.1.

Можливість автоматизувати процес тестування за допомогою спеціалізованих інструментів, підвищуючи ефективність та заощаджуючи час і ресурси	Надмірне покладання на автоматизовані інструменти може призвести до того, що вразливості можуть бути недооцінені або пропущені, що вимагатиме ручного тестування
Актуальність методики полягає у врахуванні останніх популярних загроз кібербезпеки, що відповідає сучасним викликам безпеці	Сканери вразливостей не завжди можуть виявити всі вразливості, що потребує подальшої ручної перевірки
Методологія досить гнучка, щоб застосовуватися до багатьох типів веб-сайтів, а також може поєднуватись з різними методами тестування	Методика вимагає постійного оновлення та адаптації до нових викликів, що може бути часо- та ресурсозатратним
Методика враховує стандарти доступності WCAG, забезпечуючи рівний доступ до веб-ресурсу для всіх користувачів	

Отже, з таблиці (таб. 3.1) ми можемо побачити, що розроблена методика має велику кількість переваг, що відповідають поставленим у пункті 3.1 вимогам, що робить її актуальною та всебічною для тестування веб-сайтів.

3.4 Вибір інструментів для тестування

Для проведення комплексного тестування я брала низку автоматизованих інструментів. Обраний сканер вразливостей – «HostedScan Security», інструмент для навантажувальне тестування – Apache JMeter та інструмент для тестування доступності – WAVE. Розглянемо кожен з інструментів детальніше.

- «HostedScan Security»

«HostedScan Security» є одним із інструментів для автоматизованого тестування вразливостей веб-сайтів та інших інтернет-ресурсів. Цей сервіс надає можливість сканувати веб-додатки, API, та інфраструктуру в хмарі на наявність потенційних вразливостей та конфігураційних недоліків, які можуть бути використані кіберзловмисниками для проведення атак. Повний набір сканувань сервісу включає OpenVAS, Nmap TCP та UDP, OWASP ZAP та SSLyze [22].

Розглянемо що означає кожна з цих перевірок:

OpenVAS – це повнофункціональний сканер вразливостей. Його можливості включають неавторизоване та авторизоване тестування, різні високорівневі та низькорівневі інтернет- та промислові протоколи, налаштування продуктивності для великомасштабного сканування та потужну внутрішню мову програмування для реалізації будь-якого типу тестів на вразливості. OpenVAS особливо корисний для виявлення вразливостей у мережевих сервісах, таких як веб-сервери, поштові сервери та бази даних [29].

Nmap – це потужний інструмент мережевого сканування, який може виконувати як TCP, так і UDP сканування. Сканування TCP використовується для виявлення відкритих портів в системі шляхом встановлення з'єднання з цільовим портом. UDP-сканування, з іншого боку, використовується для виявлення відкритих UDP-портів шляхом надсилання UDP-пакету на кожен цільовий порт. Сканування Nmap TCP і UDP корисно для виявлення відкритих портів, служб, запущених на цих портах, і потенційних вразливостей в мережевих службах [30].

OWASP ZAP – це сканер веб-додатків, що заснований на динамічному тестування. Він виконує серію тестів для виявлення типових вразливостей веб-додатків, таких як SQL-ін'єкції, XSS і підробка CSRF. OWASP ZAP особливо корисний для виявлення вразливостей у веб-додатках і веб-сервісах [31].

Отже, таким чином, ці сканування забезпечують комплексну оцінку стану безпеки системи шляхом виявлення багатьох видів вразливостей в мережевих службах і веб-додатках.

- Apache JMeter

Apache JMeter – це Java-додаток з відкритим вихідним кодом, що призначений для тестування функціональної поведінки та вимірювання продуктивності. Спочатку він був розроблений для тестування веб-додатків, але з тих пір розширився до інших тестових функцій. Apache JMeter можна використовувати для тестування продуктивності як статичних, так і динамічних ресурсів, веб-динамічних додатків.

З його допомогою можна імітувати велике навантаження на сервер, групу серверів, мережу або об'єкт, щоб перевірити його міцність або проаналізувати загальну продуктивність при різних типах навантаження [33].

Така універсальність дозволяє Apache JMeter ефективно тестувати різні типи систем і сервісів, що робить його комплексним інструментом для тестування і аналізу продуктивності.

- WAVE

WAVE – це інструмент оцінки веб-доступності, який допоможе оцінити доступність вашого веб-контенту, перевіривши його на відповідність Керівним принципам доступності веб-контенту (WCAG). Він надає візуальний зворотний зв'язок, розміщуючи на веб-сторінках іконки та індикатори, які вказують на проблеми з доступністю [34].

WAVE доступний як розширення для браузерів Chrome, Firefox та Edge і дозволяє безпечно оцінювати локальні, захищені паролем та конфіденційні веб-сайти. Інструмент полегшує оцінку доступності для людей та інформує користувачів про проблеми доступності. Функції WAVE включають виявлення помилок доступності, детальні звіти та пропозиції щодо усунення виявлених проблем. Це цінний інструмент для забезпечення доступності веб-контенту для людей з інвалідністю та його відповідності стандартам доступності.

4 ПРАКТИЧНА РЕАЛІЗАЦІЯ РОЗРОБЛЕНОЇ МЕТОДИКИ

4.1 Тестування веб-сайту з використанням розробленої методики

Перед початком тестування необхідно отримати відповідні дозволи та угоди від власників веб-сайтів, що забезпечує дотримання етичних норм. Для проведення тестування я обрала проектну модель веб-сайту Харківського національного університету імені В. Н. Каразіна, попередньо отримавши згоду від викладача та декана факультету комп'ютерних наук.

Отже, об'єктом проведення комплексної перевірки буде наступний веб-сайт: http://46.4.23.20/.

4.1.1 Тестування вразливостей

Отже, розпочнемо тестування з виявлення вразливостей за допомогою вищезгаданого веб-сканера «HostedScan Security».

Для початку проведення процесу тестування у браузері я відкриваю посилання на сканер: https://hostedscan.com/. Обраний мною веб-сканер має інтуїтивно зрозумілий інтерфейс. Тому для початку тестування у відповідне вікно я вводжу посилання на веб-сайт, який я хочу просканувати, а також пошту, на яку я отримаю результати (рис. 4.1).

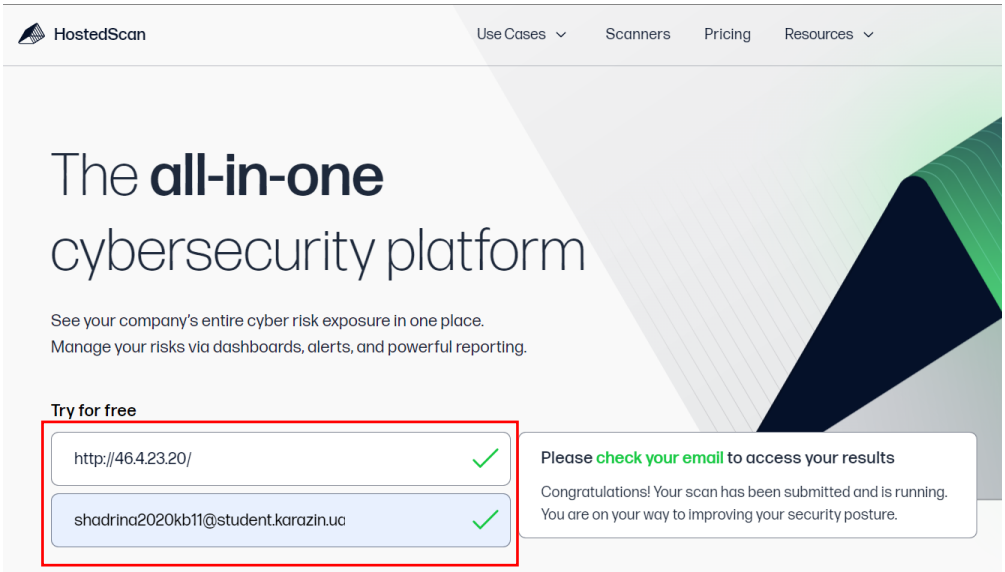


Рисунок 4.1 – Початок тестування

Далі я отримую лист на пошту із посиланням на сканування веб-сайту. Натискаю на кнопку «See results» та переходжу на сторінку з аналізом (рис. 4.2).

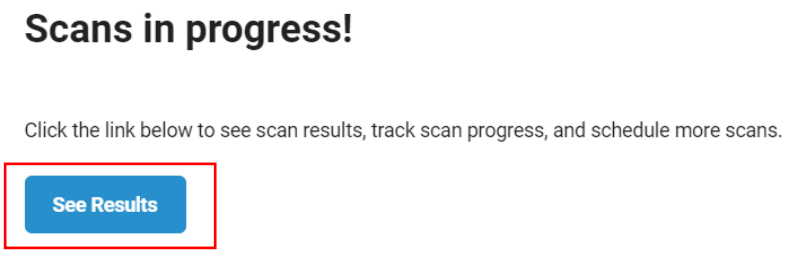


Рисунок 4.2 – Проведення тестування

Весь процес сканування обраного веб-сайту зайняв 38 хвилин. Після очікування формується звіт, який включає в себе наступні параметри:

- «Risks detected» – категорії ризиків безпеки, які були виявлені під час сканування (рис 4.3);
- «Recent Scans» – список останніх сканувань безпеки, які були проведені за допомогою цього інструменту (рис 4.4);
- «Recent Risks» – перелік найновіших виявлених ризиків (рис. 4.5);
- «Open Risks» – ризики та вразливості, які були ідентифіковані під час сканування безпеки, але ще не були усунені (рис. 4.6);
- «Exposure Window» – період часу, протягом якого система або ресурс залишається вразливим до потенційних загроз або атак (рис. 4.7).

Рисунок 4.3 – Параметр «Risks detected»

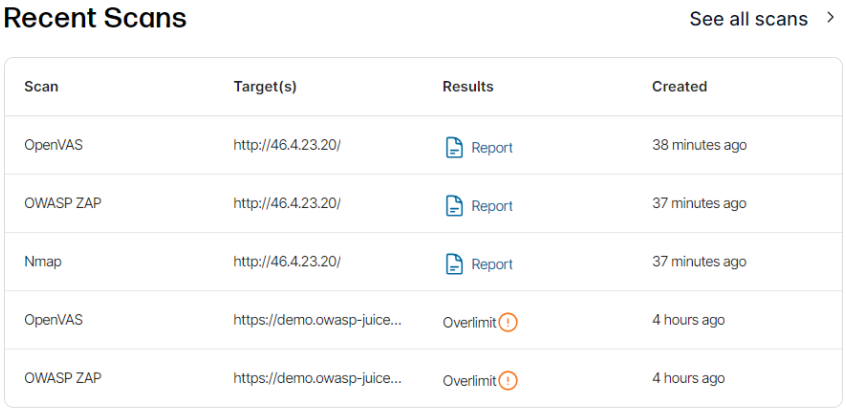


Рисунок 4.4 – Параметр «Recent Scans»

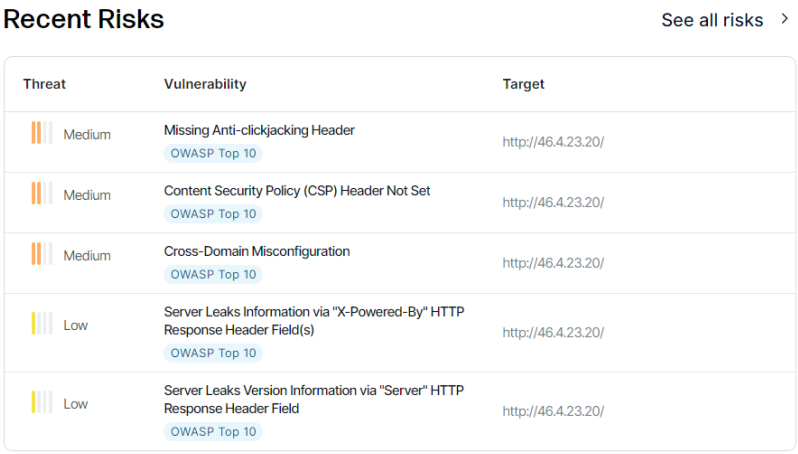


Рисунок 4.5 – Параметр «Recent Risks»

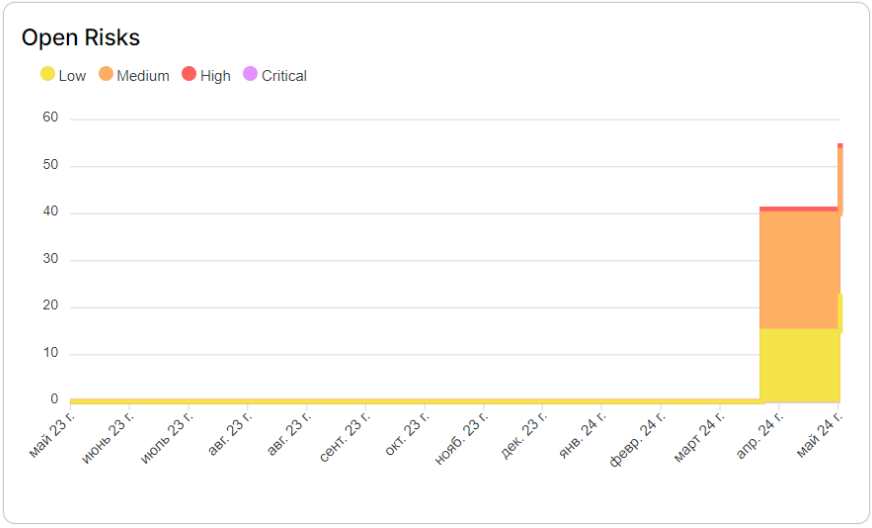


Рисунок 4.6 – Параметр «Open Risks»

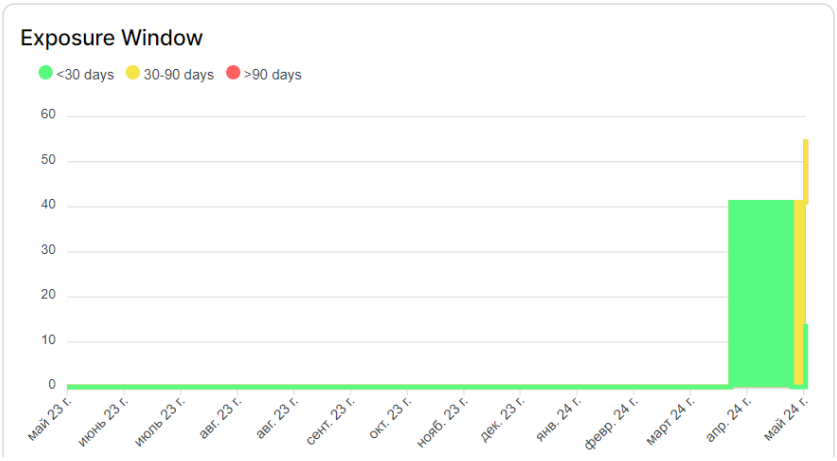


Рисунок 4.7 – Параметр «Exposure Window»

Параметр «Recent Scans» надає три детальні звіти з аналізом кожного виявленого ризику від інструментів OpenVAS, OWASP ZAP та Nmap (рис. 4.8).

Scan	Target(s)	Results
OpenVAS	http://46.4.23.20/	Report
OWASP ZAP	http://46.4.23.20/	Report
Nmap	http://46.4.23.20/	Report

Рисунок 4.8 – Параметр «Exposure Window»

Кожен звіт можна завантажити у форматі .pdf, проаналізувати кожен виявлений ризик та надати рекомендації щодо його усунення.

4.1.2 Навантажувальне тестування

Для проведення навантажувального тестування я обрала інструмент Apache JMeter. Для початку необхідно інсталювати його з веб-сайту Apache JMeter. Для своєї роботи я використовувала версію 5.5.

В Apache JMeter, компонент HTTP Request використовується для створення запитів до сервера, імітуючи дії користувача на веб-сайті. HTTP Request GET – це налаштування цього компонента для надсилання HTTP GET запиту до певного URL. HTTP GET запити зазвичай використовуються для запиту даних із сервера.

Для роботи я створила три тестових сценарії: HTTP GET запити одночасно від 10, 100 та 1000 користувачів. Таке навантаження на сервер допоможе оцінити, наскільки він є стійким до великої кількості одночасних запитів користувачів в реальних умовах роботи.

Почнемо роботу з тесту на 10 користувачів.

Для початку відкриваю додаток Apache JMeter та створюю свій тест «LoadTest» (рис 4.9):

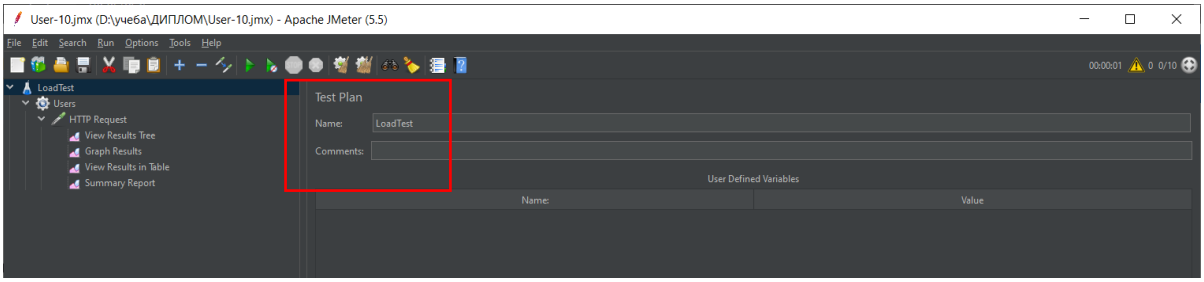


Рисунок 4.9 – Створення тесту «LoadTest»

Далі правою кнопкою мишки натискаю на створений «LoadTest» та створюю групу користувачів (рис 4.10):

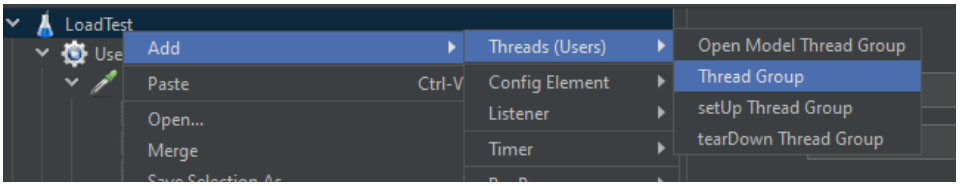


Рисунок 4.10 – Створення групи користувачів

Після чого створюю групу у 10 користувачів для першого тестового сценарію (рис. 4.11):

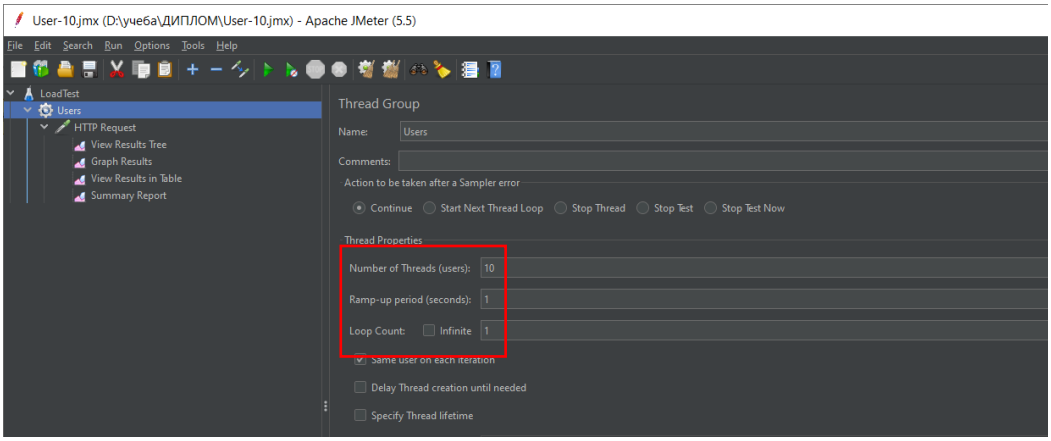


Рисунок 4.11 – Створення групи у 10 користувачів

Далі створюю HTTP запит для запуску тестового сценарію (рис. 4.12):

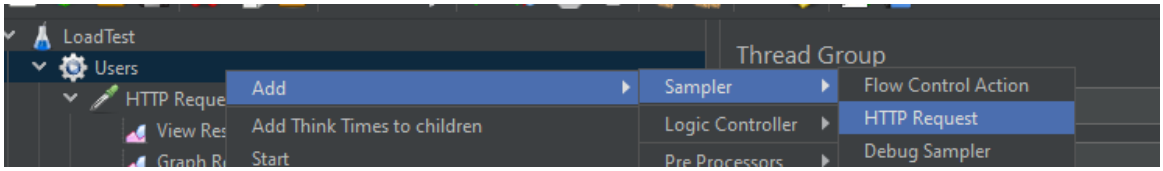


Рисунок 4.12 – Створення HTTP запиту

Обираю запит GET та вставляю посилання на досліджуваний сайт (рис. 4.13):

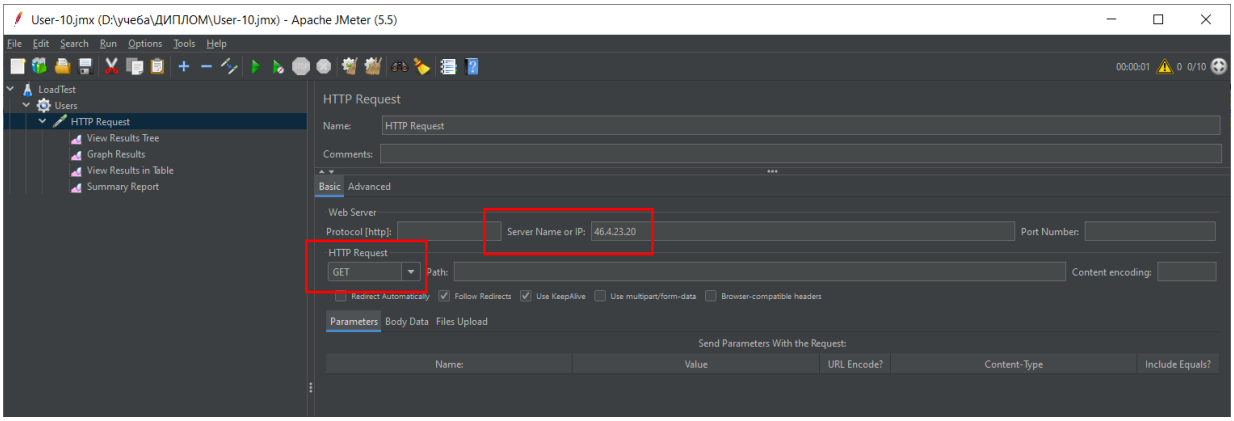


Рисунок 4.13 – Створення HTTP GET запиту

Перед запуском тесту також обираю опції, завдяки яким зможу проаналізувати результат (рис. 4.14):

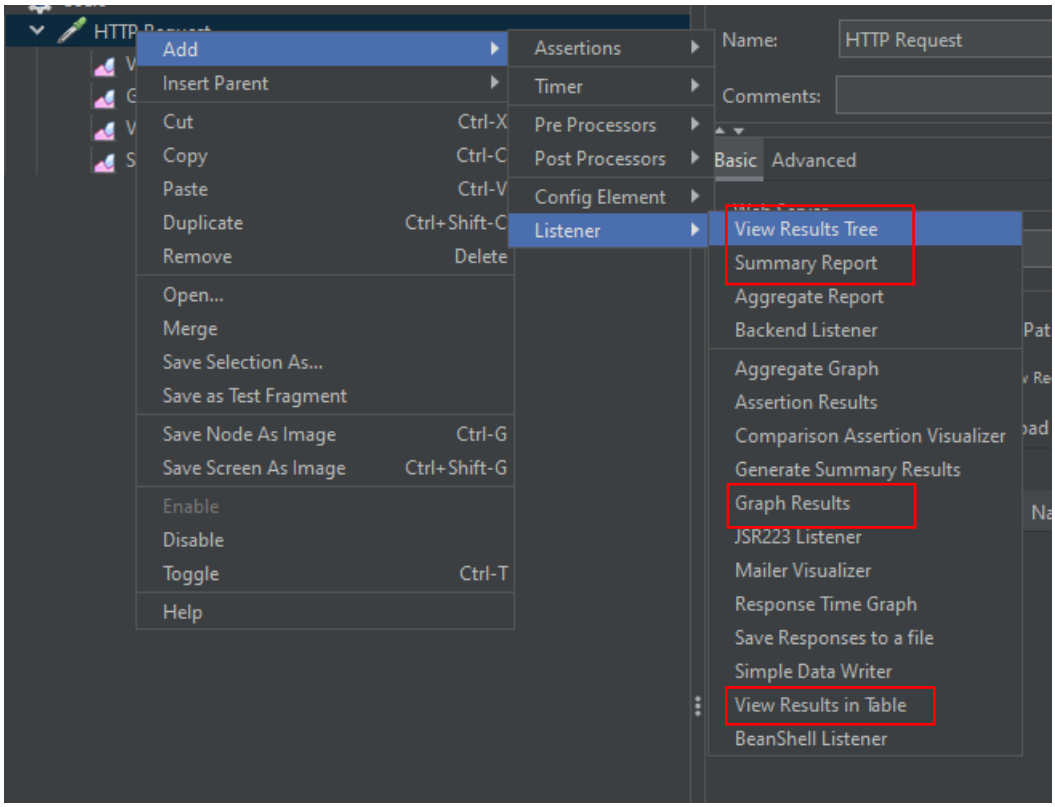


Рисунок 4.14 – Вибір опцій для отримання результату тестування

Для аналізу я обрала наступні критерії:

- View Results Tree відображає деревоподібне представлення всіх результатів вибірки, включно з деталями запиту, даними відповіді і будь-якими пов'язаними підвибірками.
- Summary Report генерує зведений звіт про результати тестування, включаючи статистику, таку як кількість зразків, середній час відгуку, пропускну здатність і частоту помилок.
- Graph Results генерує графічні представлення різних показників продуктивності, таких як час відгуку, пропускна здатність і помилки, протягом виконання тесту.
- View Results in Table відображає результати тесту в табличному форматі, дозволяючи сортувати і фільтрувати дані за потребою.

Запустивши перший тестовий сценарій отримую наступні результати (метрики View Results Tree та View Results in Table наявні у додатку Б):

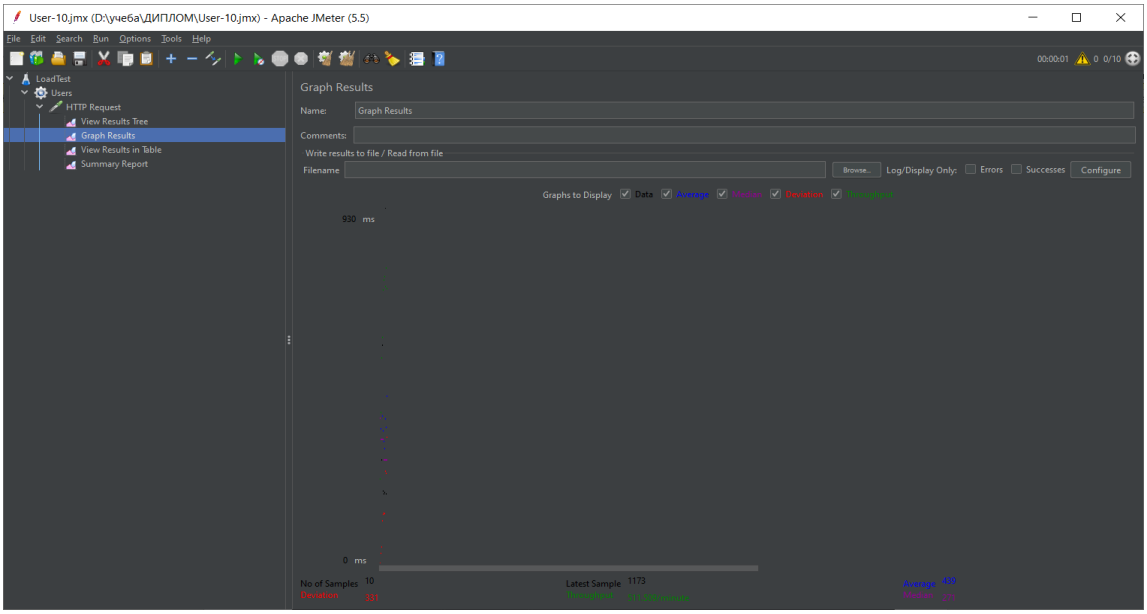


Рисунок 4.15 – Graph Results

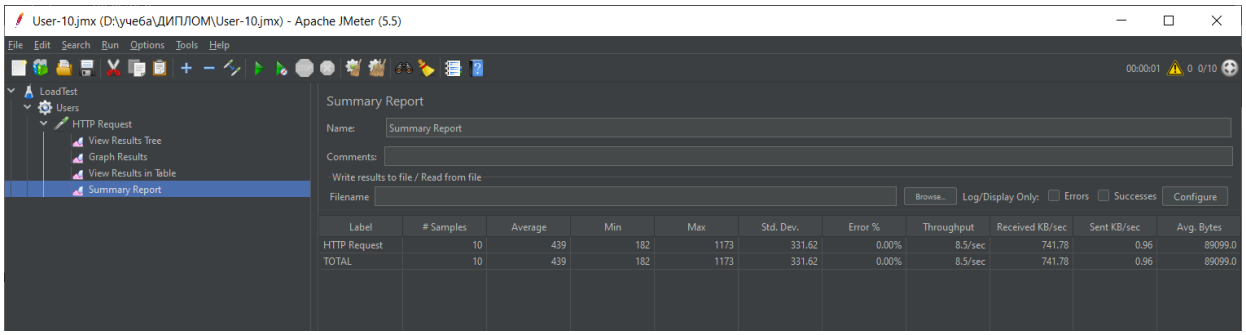


Рисунок 4.16 – Summary Report

Аналогічним чином виконую тестовий сценарій на 100 користувачів:

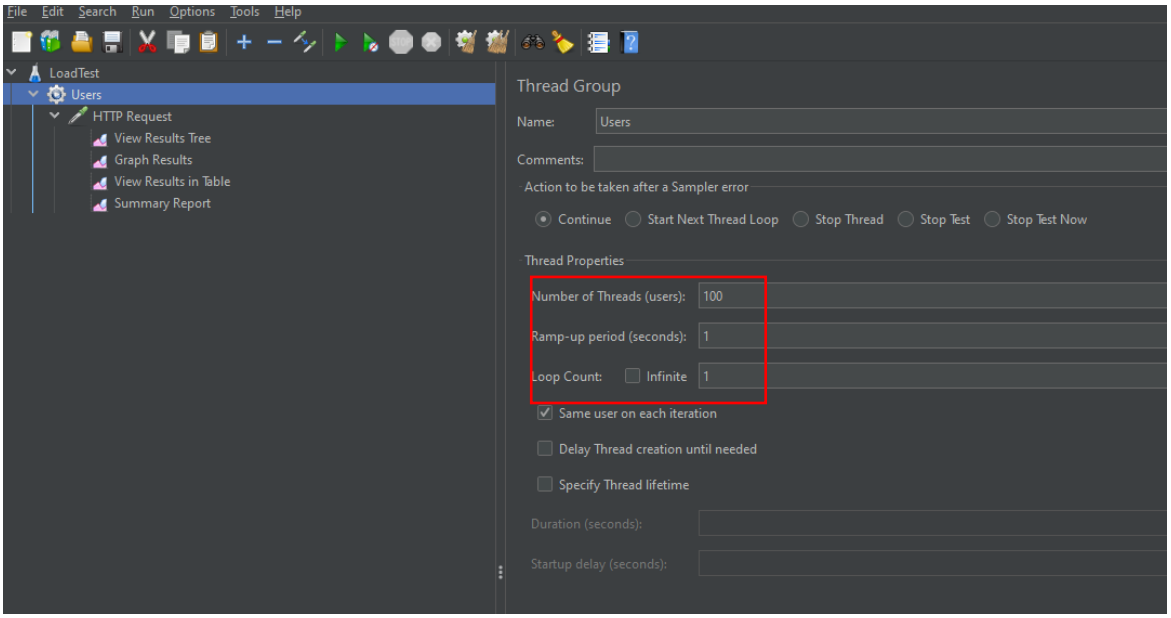


Рисунок 4.17 – Створення групи у 100 користувачів

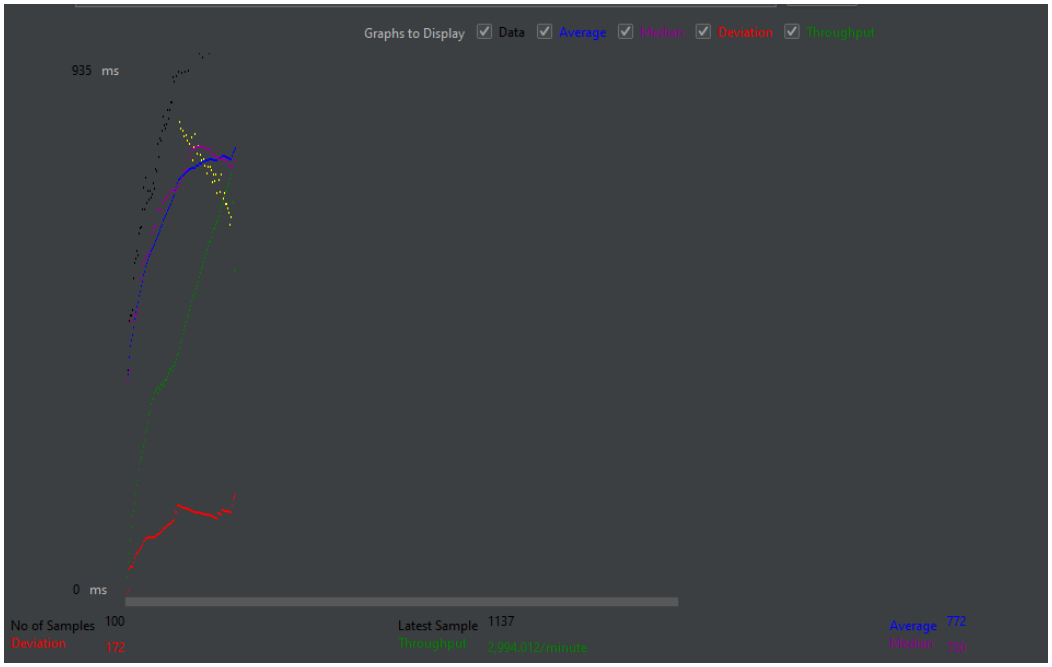
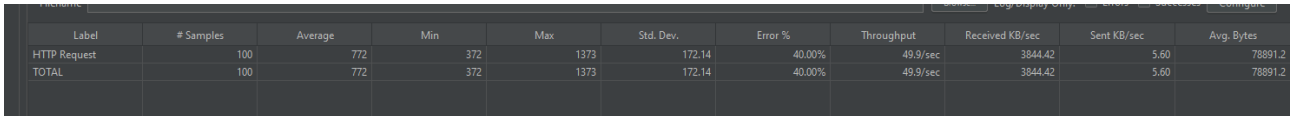


Рисунок 4.18 – Graph Results



Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughput	Received KB/sec	Sent KB/sec	Avg. Bytes
HTTP Request	100	772	372	1373	172.14	40.00%	49.9/sec	3844.42	5.60	78891.2
TOTAL	100	772	372	1373	172.14	40.00%	49.9/sec	3844.42	5.60	78891.2

Рисунок 4.19 – Summary Report

Аналогічно виконую тестовий сценарій для 1000 користувачів:

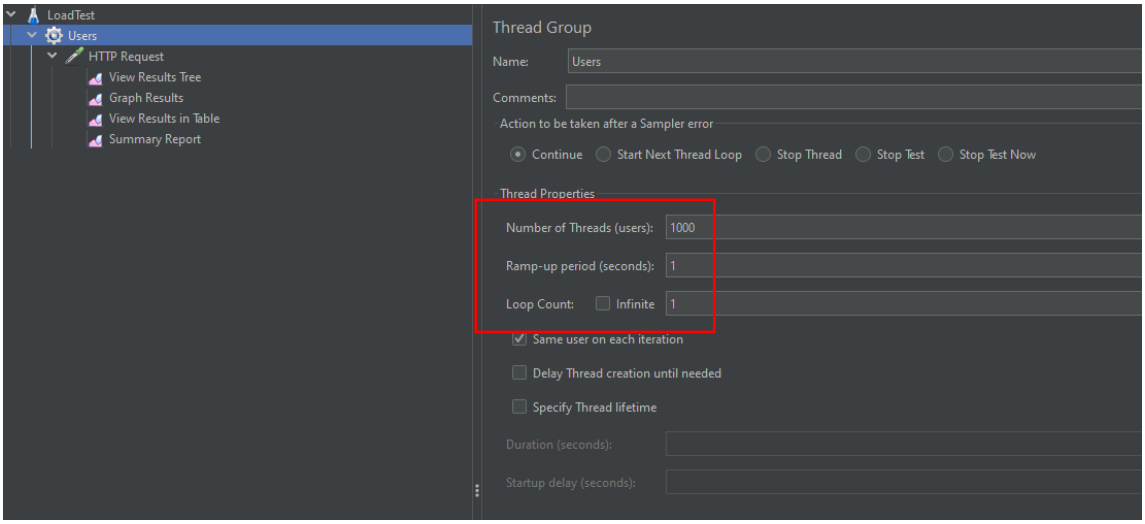


Рисунок 4.20 – Створення групи у 1000 користувачі

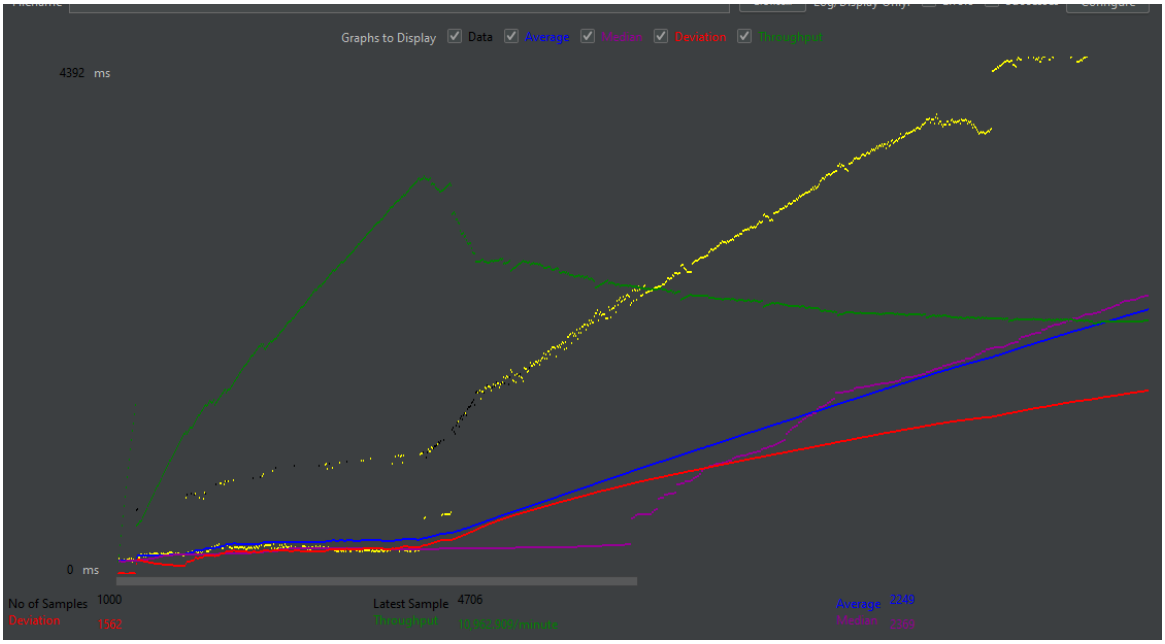


Рисунок 4.21 – Graph Results

Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughput	Received KB/sec	Sent KB/sec	Avg. Bytes
HTTP Request	1000	2249	109	4810	1562.60	93.80%	182.7/sec	2755.33	15.18	15441.9
TOTAL	1000	2249	109	4810	1562.60	93.80%	182.7/sec	2755.33	15.18	15441.9

Рисунок 4.22 – Summary Report

4.1.3 Тестування доступності

Останнім етапом у тестуванні є тестування доступності веб-сайту. Для цього я використаю інструмент WAVE, попередньо встановивши це розширення на браузер Google Chrome.

Для початку тестування клікаю правою кнопкою мишки на відкритій сторінці сайту та у спилавючому вікні та обираю пункт «WAVE this page» (рис. 4.23).

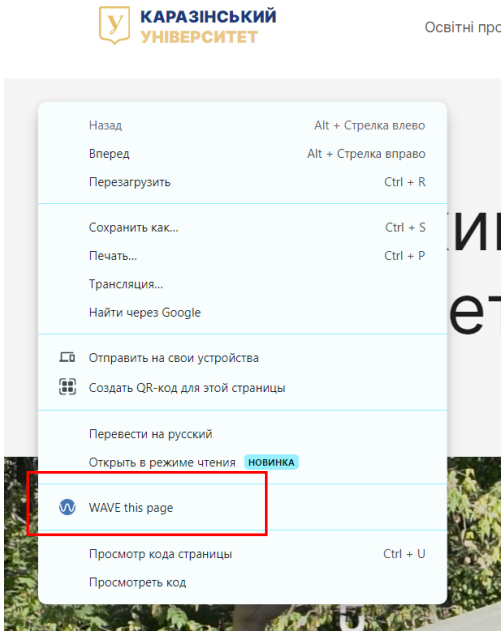


Рисунок 4.23 – «WAVE this page»

Після цього ми бачимо кількість кожного виду помилок, які були виявлені (рис. 4.24), а також де саме сайті була виявлена та чи інша помилка (рис. 4.25).

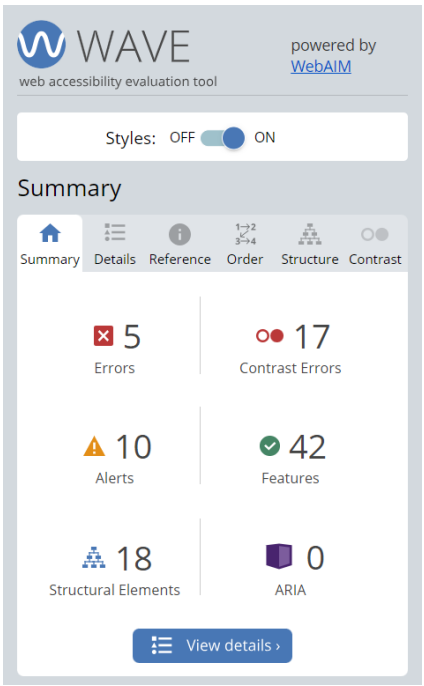


Рисунок 4.24 – Наданий звіт

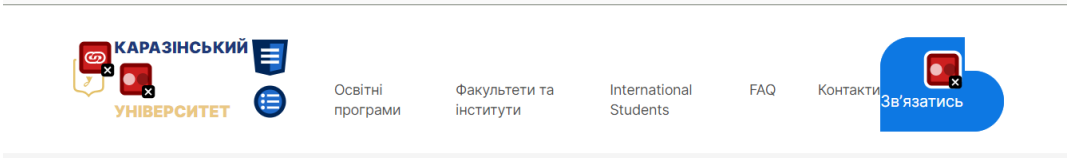


Рисунок 4.25 – Наданий звіт

Якщо перейти на вкладку «Details» отримую всі детальні помилки, натиснувши на які додаток показує, де саме ця помилка знаходиться на сайті для зручності (рис. 4.26 - 4.27). Також, натиснувши на маленьку літеру «i», бачимо пояснення, що це за помилка, що вона означає та як її виправити (рис. 4.28).

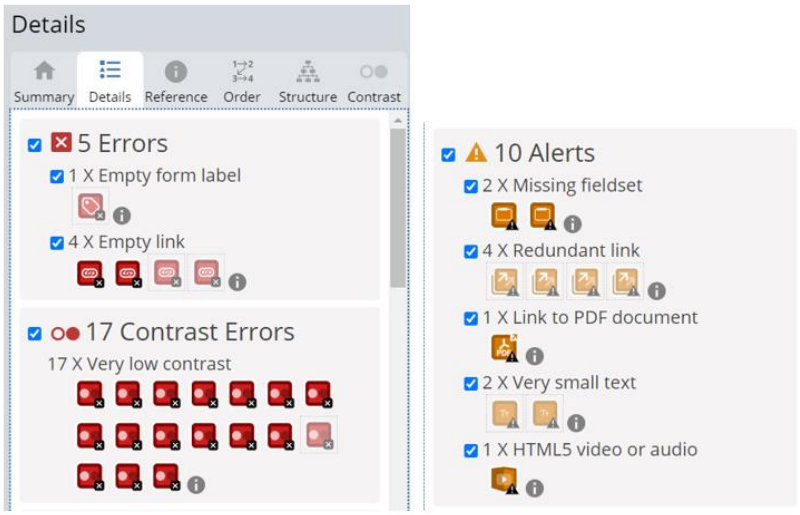


Рисунок 4.26 – Детальні аналіз

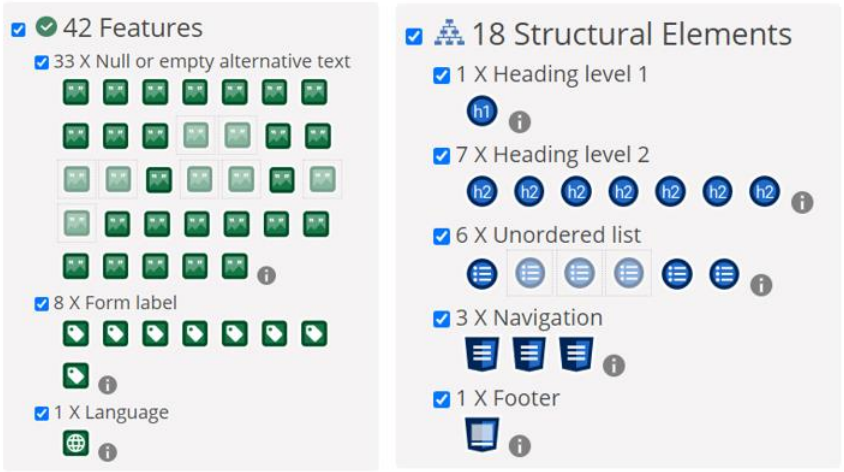


Рисунок 4.27 – Детальний аналіз

Як видно з рис. 4.26 та 4.27 всього було виявлено 5 помилок, 17 контрастних помилок, 10 попереджень, 42 особливості та 18 структурних елементи.

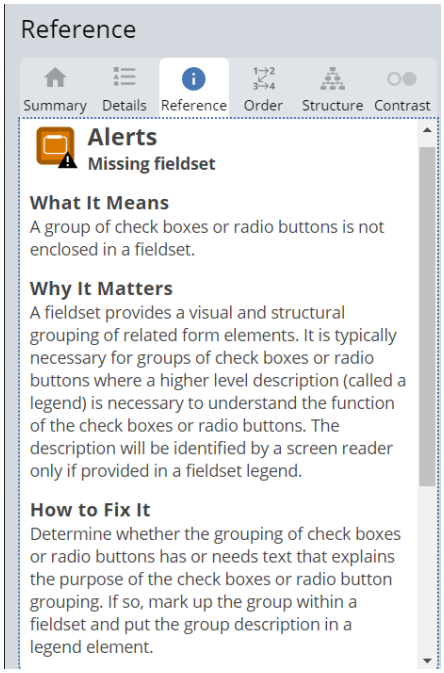


Рисунок 4.28 – Детальний аналіз

Розглянемо всі види помилок, попереджень особливостей та структурних елементів, які були виявлені:

- Empty form label

Мітка форми присутня, але не містить жодного вмісту. Це важливо тому, що елемент <label>, пов’язаний із елементом керування форми, але не містить тексту, не надасть користувачеві жодної інформації про елемент керування форми.

- Empty link

Посилання не містить тексту. Якщо посилання не містить тексту, функція чи призначення посилання не будуть представлені користувачеві.

- Very Low Contrast

Дуже низький контраст між кольорами тексту та фону. Достатня контрастність тексту необхідна для всіх користувачів, особливо для людей зі слабким зором.

- Missing fieldset

Група прапорців або перемикачів не вкладається в набір полів. Набір полів забезпечує візуальне і структурне групування пов'язаних елементів форми. Зазвичай він необхідний для груп прапорців або перемикачів, коли для розуміння

функцій прапорців або перемикачів потрібен опис вищого рівня. Опис буде ідентифікований програмою для зчитування з екрана, тільки якщо він міститься в легенді набору полів.

- Redundant link

Суміжні посилання ведуть на ту саму URL-адресу. Коли суміжні посилання ведуть в одне і те ж місце, це призводить до додаткової навігації і повторень для користувачів клавіатурних і екранних зчитувачів.

- Link to PDF document

Посилання на документ у форматі PDF. Якщо PDF-документи не були створені з урахуванням вимог доступності, у них часто виникають проблеми з доступністю. PDF-документи зазвичай переглядаються за допомогою окремої програми або плагіна, що може спричинити плутанину та труднощі з навігацією.

- Very small text

Дуже маленький текст. Дуже дрібний текст важко читати, особливо людям зі слабким зором.

- HTML5 video or audio

Присутні відео або аудіо елементи. Аудіоконтент повинен бути представлений у текстовому форматі, щоб бути повністю доступним для глухих і слабочуючих користувачів. Відеоконтент зі звуком повинен мати синхронізовані субтитри та транскрипт.

- Null or empty alternative text

Альтернативний текст є нульовим або порожнім. Якщо зображення не передає зміст або якщо зміст зображення передається в іншому місці, зображення повинно мати порожній/нульовий альтернативний текст, щоб гарантувати, що воно ігнорується програмою для зчитування з екрана і ховається, коли зображення вимкнені або недоступні.

- Form label

Мітка форми присутня і пов'язана з елементом управління форми. Належним чином пов'язана мітка форми відображається користувачеві програми для читання з екрана, коли він отримує доступ до елемента керування формою.

- Language

Визначається мова документа або елемента сторінки. Визначення мови сторінки або частини сторінки дозволяє програмам для зчитування з екрана читати вміст належним чином.

- Heading level 1

Присутній заголовок першого рівня. Заголовки полегшують навігацію сторінками для користувачів допоміжних технологій. Вони також надають документу смислове та візуальне значення і структуру. Заголовки першого рівня повинні містити найважливіший заголовок на сторінці.

- Heading level 2

Присутній заголовок другого рівня. Заголовки полегшують навігацію сторінками для користувачів допоміжних технологій. Вони також надають документу смислове та візуальне значення і структуру.

- Unordered list

Присутній невпорядкований список. Невпорядковані списки представляють групу пов'язаних, паралельних елементів. Користувачі багатьох допоміжних технологій можуть здійснювати навігацію за списками та всередині них.

- Navigation

Присутній навігаційний орієнтир. Навігація ідентифікує розділ навігаційних посилань і може полегшити семантику сторінки та навігацію по сторінці.

- Footer

Присутній колонтитул або контент-інформаційна мітка. Колонтитули визначають нижній колонтитул сторінки або розділу сторінки. Зазвичай вони вказують на авторство, пов'язані посилання, дату копірайту або інший вміст нижнього колонтитулу. Колонтитули полегшують семантику сторінки та навігацію.

4.2 Аналіз результатів тестування та рекомендації щодо усунення виявлених недоліків

Почнемо аналіз результатів проведеного комплексного тестування зі звіту, який надав сканер вразливостей «HostedScan Security».

Всього було виявлено 14 ризиків: 6 – середніх та 8 – низьких (діаграма 4.1):

Діаграма 4.1 – Кількість виявлених ризиків

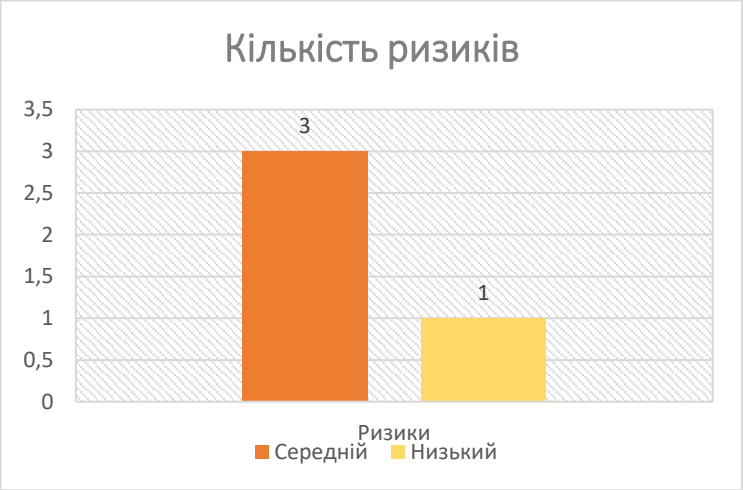
Ризики середнього рівня можуть мати помірний вплив на безпеку системи. Їх також потрібно виправити, але вони можуть не бути настільки критичними, як високі ризики.

Низькі ризики вказують на менш критичні проблеми, які можуть мати обмежений вплив на безпеку системи. Вони також потребують уваги, але вони менш нагальні на не потребують негайного виправлення.

Отже, загальна кількість виявлених ризиків та їх рівні показують широкий спектр потенційних проблем з безпекою, які потребують уваги та вжиття заходів для забезпечення надійності та захищеності системи.

Розглянемо декілька ризиків, які виявили під час сканування Nmap, OWASP ZAP та OpenVAS.

Інструмент Nmap виявив 3 середніх ризики та 1 низький (діаграма 4.2):



Діаграма 4.2 – Кількість виявлених ризиків інструментом Nmap

Перелік всіх виявлених ризиків зображено на рис. 4.29:

Title	Severity	Open	Accepted
Open TCP Port: 2244	Medium	1	0
Open TCP Port: 2236	Medium	1	0
Open TCP Port: 34568	Medium	1	0
Open TCP Port: 80	Low	1	0

Рисунок 4.29 – Перелік всіх виявлених Nmap вразливостей

Розглянемо детально ризик Open TCP Port: 2244.

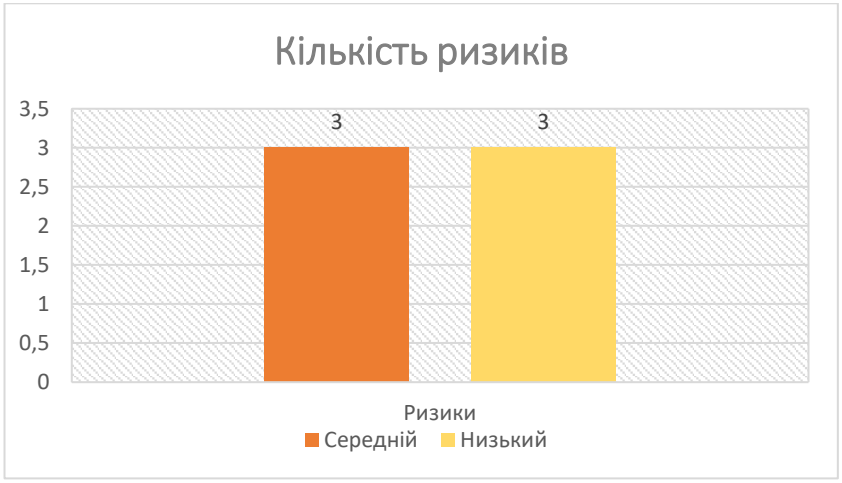
Опис: відкритий порт може бути очікуваною конфігурацією. Наприклад, веб-сервери використовують порт 80 для обслуговування веб-сайтів через http і порт 443 для обслуговування веб-сайтів за протоколом https. Несподівано відкритий порт може надати ненавмисний доступ до програм, даних і приватних мереж. Відкриті порти також можуть бути небезпечними коли очікувані сервіси застаріли і використовуються через вразливості в системі безпеки.

Рішення: важливо перевірити, чи порт 2244 є дійсно необхідним для роботи системи. Якщо він не використовується, рекомендується закрити його для зменшення потенційних вразливостей.

Якщо цей порт дійсно використовується необхідно переконатись, що ПЗ, яке використовує цей порт, має всі необхідні оновлення для усунення вразливостей. Також слід застосовувати заходи безпеки (брандмауери, VPN, багатофакторна аутентифікація та регулярне оновлення системи).

Інші вразливості розглянуті у додатку А.

Інструмент OWASP ZAP виявив 3 середніх ризики та 3 низьких (діаграма 4.3):



Діаграма 4.3 – Кількість виявлених ризиків інструментом OWASP ZAP

Перелік всіх виявлених ризиків зображено на рис. 4.30:

Passive Web Application Vulnerabilities	Severity	First Detected	Last Detected
Cross-Domain Misconfiguration	Medium	0 days ago	0 days ago
Missing Anti-clickjacking Header	Medium	0 days ago	0 days ago
Content Security Policy (CSP) Header Not Set	Medium	0 days ago	0 days ago
X-Content-Type-Options Header Missing	Low	0 days ago	0 days ago
Server Leaks Information via "X-Powered-By" HTTP Response Header Field(s)	Low	0 days ago	0 days ago
Server Leaks Version Information via "Server" HTTP Response Header Field	Low	0 days ago	0 days ago

Рисунок 4.30 – Перелік всіх виявлених OWASP ZAP вразливостей

Розглянемо детально ризик Cross-Domain Misconfiguration.

Опис: можливе завантаження даних веб-браузера через неправильну конфігурацію CORS на веб-сервері.

*Рішення: н*еобхідно переконатись, що конфіденційні дані не доступні без автентифікації.

Налаштуйте HTTP-заголовок «Access-Control-Allow-Origin» на більш обмежений набір доменів або повністю видаліть усі заголовки CORS, щоб дозволити веб-браузеру застосовувати політику однакового походження (SOP) у більш обмежувальний спосіб.

Інші вразливості розглянуті у додатку А.

Інструмент OpenVAS виявив 4 низьких ризики. Перелік всіх виявлених ризиків зображено на рис. 4.31:

Network Vulnerabilities	Severity	First Detected	Last Detected
TCP Timestamps Information Disclosure cvss score: 2.6	Low	0 days ago	0 days ago
Weak MAC Algorithm(s) Supported (SSH) cvss score: 2.6	Low	0 days ago	0 days ago
Weak MAC Algorithm(s) Supported (SSH) cvss score: 2.6	Low	0 days ago	0 days ago
ICMP Timestamp Reply Information Disclosure cvss score: 2.1	Low	0 days ago	0 days ago

Рисунок 4.31 – Перелік всіх виявлених OpenVAS вразливостей

Розглянемо ризик Weak MAC Algorithm(s) Supported (SSH).

Опис: віддалений SSH-сервер налаштовано на дозвіл/підтримку слабких MAC алгоритмів.

Рішення: необхідно вимкнути алгоритми слабких MAC-адрес, про які повідомляється.

Інші вразливості розглянуті у додатку А.

Отже, з першого етапу тестування можна зробити висновок, що загальна кількість виявлених ризиків та їх рівні показують, що система виявила обмежену кількість потенційних проблем з безпекою, які потребують уваги та вжиття відповідних заходів для забезпечення надійності та захищеності системи. Це означає, що веб-сайт є стійким до потенційних атак зловмисників.

Продовжимо аналіз проведеного тестування огляду навантажувального тестування, який я провела за допомогою інструмента Apache JMeter.

У першому тестовому сценарії загалом було виконано 10 одночасних запитів на сервер. Аналізуючи результати, видно, що всі запити пройшли без помилок. Максимальний час відгуку становить 1173 мс, що є прийнятним для низького навантаження. З графіку видно, що запити виконувались без значних піків та помилок, час відгуку системи залишається відносно стабільним і низьким протягом усього тесту. Відсоток помилок – 0% та обсяг переданих даних – 0,96 Кб/сек свідчать про гарну продуктивність при низькому навантаженні.

У другому тестовому сценарії загалом було виконано 100 одночасних запитів на сервер. Аналізуючи результати, видно, що в даному випадку не всі запити пройшли без помилок, загальний відсоток – 40%. На другому графіку з

середнім навантаженням спочатку час відгуку зростає швидко, але потім стабілізується на певному рівні після короткого періоду нестабільності. Перші 52 запити пройшли, після 53 вже починаються проблеми з проходженням запитів. Максимальний час відгуку становить 1373 мс, а обсяг переданих даних – 5,60 Кб/сек свідчать про середню продуктивність при помірному навантаженні.

У третьому тестовому сценарії загалом було виконано 1000 одночасних запитів на сервер. Третій графік з високим навантаженням демонструє значне зростання часу відгуку з плином часу, що свідчить про перевантаження системи та деградацію продуктивності при обробці великої кількості запитів одночасно. В даному випадку вже дуже високий відсоток помилок – 93,80%. Це свідчить про те, що лише 6% відсотків від загальної кількості запитів пройшли успішно, що є дуже поганим результатом. Максимальний час відгуку становить 4810 мс, а обсяг переданих даних – 15,18 Кб/сек свідчать про дуже низьку продуктивність при високому навантаженні в 1000 користувачів.

Для того, щоб наочно побачити та порівняти результати складемо порівняльну таблицю (таб. 4.1).

Таблиця 4.1 – Порівняльний аналіз результатів

Метрика	10 юзерів	100 юзерів	1000 юзерів
Кількість запитів	10	100	1000
Відсоток помилок	0%	40,00%	93,80%
Середній час відгуку	439 мс	772 мс	2249 мс
Максимальний час відгуку	1173 мс	1373 мс	4810 мс
Пропускна здатність	8,5/сек	49,9/сек	182,7/сек
Обсяг переданих даних	0,96 КБ/сек	5,60 Кб/сек	15,18 КБ/сек

З таблиці 4.1 видно, що з підвищенням кількості користувачів час відгуку збільшується. Наприклад, середній час відгуку при 1000 користувачах становить близько 2249 мс, що вказує на відчутну затримку. Пропускна здатність також зменшується зі збільшенням навантаження. При 1000 користувачах пропускна здатність близько 1827 запитів на секунду, що значно менше, ніж при 10 користувачах. Особливо різницю видно, якщо порівнювати відсотки помилок,

якщо при 10 користувачх ми маємо повністю успішні запити, то при 1000 – 93,8% помилок, що є дуже поганим результатом.

Отже, з усіх даних можна зробити висновок, що система показує гарні результати лише при низькому навантаженні на неї у 10 користувачів.

Для того, щоб покращити стійкість системи та забезпечити більш високу ефективність при обслуговуванні великої кількості одночасних користувачів можна використовувати деякі з наведених нижче стратегій.

По-перше, необхідно переконатись, що серверне ПЗ налаштовано для оптимальної роботи при високому навантаженні на нього. Для цього необхідно проаналізувати і оптимізувати критичні компоненти системи, такі як бази даних, кешування, обробка запитів тощо.

Ще одним варіантом вирішення проблеми може бути встановлення систем моніторингу продуктивності для раннього виявлення високого навантаження та автоматичного масштабування для забезпечення стабільної роботи під навантаженням задля вирішення проблеми до того, як велика кількість запитів призведе до серйозних збоїв у роботі системи.

Також для розподілу запитів між декількома серверами можна використовувати балансувальники навантаження, що може допомогти зменшити час відгуку і збільшити загальну пропускну здатність системи або використовувати розподілену архітектуру для горизонтального масштабування.

Регулярне проведення навантажувального тестування та аналіз отриманих результатів допоможе оптимізувати систему, щоб підвищити стійкість до високих навантажень на неї.

Останнім кроком в опрацюванні результатів є аналіз результатів тестування доступності. Розглянемо по одній з кожних виявлених помилок та попереджень, а також надамо вирішення цієї проблеми. Детальний аналіз всіх знайдених проблем наведено у додатку Б.

Першою виявленою помилкою є «Empty form label» (рис. 4.32):

Рисунок 4.32 – Помилка «Empty form label»

Ця помилка означає, що мітка форми присутня, але не містить жодного вмісту. І це дійсно так, якщо натиснути на ці три риски, які видно лише не в повноекранному режимі, то там дійсно пусто.

Щоб виправити цю помилку, необхідно додати відповідний текст до цього спливаючого меню. Підписи не потрібні тільки для елементів управління зображеннями, редагування, скидання, кнопок або прихованих форм.

Одна з виявлених контрастних помилок зображена на рис. 4.33:

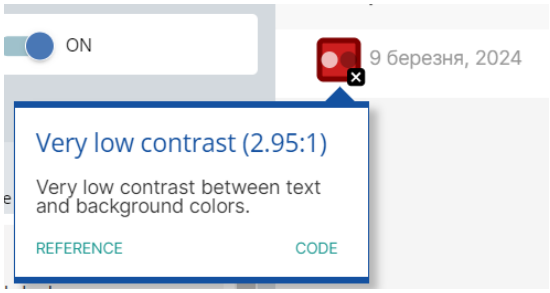


Рисунок 4.33 – Помилка «Very low contrast»

Ця помилка означає, що контраст між текстом і фоном дуже низький (2,95:1), що може заважати сприйняттю написаного людьми з обмеженими можливостями. WCAG вимагає, щоб для елементів сторінки були визначені кольори переднього і заднього планів, які забезпечують достатню контрастність. WCAG Рівень AAA вимагає контрастності щонайменше 7:1 для звичайного тексту та 4,5:1 для великого тексту (більше 18 пунктів або 14 пунктів жирним шрифтом). Присутній текст з контрастністю менше 4,5:1, тому щоб це виправити необхідно збільшити контраст між кольором тексту і кольором фону.

Далі розглянемо одне з попереджень на рис. 4.34:

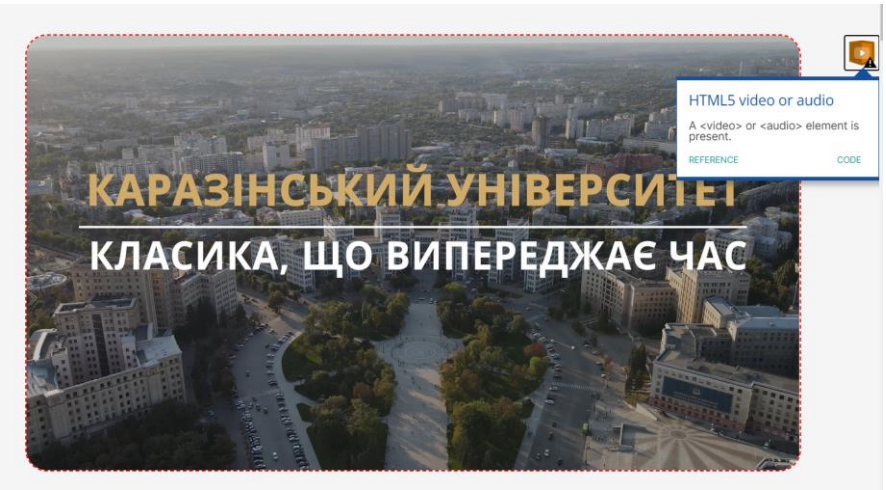


Рисунок 4.34 – Попередження «HTML5 video or audio»

Це означає, що на сайті присутній відео або аудіо елемент. Аудіоконтент повинен бути представлений у текстовому форматі, щоб бути повністю доступним для глухих і слабочуючих користувачів. Відеоконтент зі звуком повинен мати синхронізовані субтитри. В нашому випадку, це попередження виправляти не потрібно, адже воно містить в собі синхронізовані субтитри.

Отже, щодо тестування доступності, веб-сайт показує хороші результати, зважаючи на те, що більшість з виявлених проблем – це контрастні помилки, які легко виправити просто збільшивши контраст між текстом і фоном. Розглянемо короткий висновок щодо повного аналізу проведеного комплексного тестування розробленою методикою.

a) Результати тестування безпеки:

Всього виявлено 14 ризиків (6 середніх та 8 низьких). Це свідчить про наявність потенційних вразливостей, що можуть вплинути на безпеку системи, проте відсутність високорівневих ризиків може вказувати на загально задовільний стан безпеки.

b) Результати навантажувального тестування:

При низькому навантаженні (10 користувачів) система показує відмінні результати без помилок, що свідчить про її здатність ефективно працювати при малих навантаженнях.

При середньому навантаженні (100 користувачів) з’являються помилки (40% запитів), що вказує на потребу оптимізації конфігурації сервера або ПЗ.

При високому навантаженні (1000 користувачів) велика кількість помилок (93,80%) та суттєве збільшення часу відгуку свідчать про недостатність ресурсів системи для роботи при високих навантаженнях.

Отже, результати показують, що система ефективно працює при низькому навантаженні, але виявляє суттєві недоліки при середньому та високому навантаженні. Основні проблеми включають значне зростання часу відгуку і збільшення відсотка помилок зі зростанням кількості користувачів.

с) Результати тестування доступності:

Виявлені помилки, пов’язані з конфігурацією доступності, зокрема з контрастністю та мітками форм. Ці помилки легко виправити, що може покращити загальну доступність веб-сайту для користувачів з обмеженими можливостями.

Отже, система у випадку тестування її вразливостей сканером безпеки «HostedScan Securiry» та тестування доступності за допомогою розширення до браузеру WAVE показує цілком гарні результати, але все ще потребує ряду вдосконалень. Безпека системи на задовільному рівні через відсутність високих ризиків, але з потенційними точками для покращення, особливо у випадку середніх ризиків. Система також показує хороші результати при низьких навантаженнях, але виявляє значні проблеми при середніх та високих навантаженнях. Результати навантажувальних тестів підкреслюють потребу в оптимізації системи для забезпечення її стабільності та ефективності при різних умовах використання. Тому рекомендується найбільше уваги звернути саме на цю проблему та вжити відповідних заходів щодо оптимізації продуктивності системи.

ВИСНОВКИ

У сучасному світі, де інтернет грає важливу, можна сказати навіть центральну роль в нашому житті, безпека та продуктивність роботи веб-сайтів, якими ми користуємось кожного дня стає все більш важливою задачею. Незважаючи на те, що кожен веб-сайт має свої вразливості та недосконалості у роботі, які можуть бути використані зловмисниками для здійснення кібератак, розробка ефективних методів захисту та тестування є критично важливою для забезпечення конфіденційності, цілісності, а також доступності інформації для всіх користувачів.

У даній дипломній роботі було проведено комплексне дослідження теоретичних основ загроз та захисту веб-сайтів, розглянуто основні види атак, що використовуються зловмисниками, а також методи та інструменти захисту від кібератак. На основі аналізу існуючих методів тестування безпеки веб-ресурсів була розроблена власна методика тестування для виявлення різних видів недоліків.

Розроблена методика тестування дозволяє не тільки ідентифікувати різноманітні уразливості веб-сайтів, але й оцінювати стійкість системи під час різних рівнів навантаження та перевіряти її доступність для користувачів з обмеженими можливостями.

Основні результати дослідження показали:

При тестуванні вразливостей веб-сканером було виявлено 14 ризиків, серед яких 6 середнього рівня та 8 низького рівня. Це підкреслює необхідність проведення ретельного тестування веб-сайтів з метою своєчасного виявлення та усунення потенційних загроз, що може забезпечити захист від несанкціонованого доступу та зловмисного використання ресурсів.

Під час навантажувального тестування результати показали, що система ефективно справляється з низьким навантаженням без помилок, але при середньому та високому навантаженні відбувається значне погіршення продуктивності та збільшення часу відгуку. Це вказує на необхідність

оптимізації системи для підтримки стабільної роботи при збільшенні навантаження.

Тестування доступності показало. Що виявлені недоліки, здебільшого пов’язані з контрастністю та навігацією, підкреслюють важливість адаптації веб-інтерфейсів для забезпечення кращого доступу для людей з обмеженими можливостями. Це не тільки покращує загальну користувацьку досвід, але й забезпечує дотримання нормативних вимог щодо доступності веб-контенту.

На основі отриманих результатів у дипломній роботі були сформовані рекомендації щодо усунення виявлених недоліків. Впровадження запропонованих рекомендацій допоможе підвищити рівень кібербезпеки веб-ресурсу, забезпечити належну конфіденційність, цілісність і доступність інформаційних систем, підвищити рівень стійкості системи при високих навантаженнях, а також покращити доступність вмісту для всіх користувачів.

Отже, результати цієї роботи підтверджують актуальність розробки методів захисту та тестування вразливостей веб-сайтів. Методика, розроблена в цій роботі, може бути корисною для виявлення та усунення недоліків веб-сайтів, що допоможе забезпечити безпеку та стабільність їх роботи. Рекомендації щодо усунення виявлених ризиків можуть бути корисними для розробників веб-сайтів та спеціалістів з кібербезпеки.

У зв'язку з постійним розвитком та ускладненням методів кібератак, ця робота є важливим внеском у розвиток методів захисту та тестування вразливостей веб-сайтів. Вона може бути корисною для фахівців, які працюють над забезпеченням безпеки веб-сайтів, а також для тих, хто розробляє нові методи захисту від кібератак.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Що таке кібербезпека [Електронний ресурс]. – Режим доступу: https://www.microsoft.com/uk-ua/security/business/security-101/what-is-cybersecurity.
2. Практичні завдання з тестування програмного забезпечення [Електронний ресурс] / КПІ ім. Ігоря Сікорського. – Режим доступу: https://ela.kpi.ua/server/api/core/bitstreams/bc0307b0-b4c8-491d-adef-bee458c2a660/content.
3. OWASP Top 10 [Електронний ресурс]. – Режим доступу: https://owasp.org/www-project-top-ten/
4. Analysis of vulnerabilities and security problems of web applications [Електронний ресурс] / ResearchGate. – Режим доступу: https://www.researchgate.net/publication/373082411_Analysis_of_vulnerabilities_and_security_problems_of_web_applications.
5. Аналітика [Електронний ресурс] / PT Security. – Режим доступу: https://www.ptsecurity.com/ru-ru/research/analytics/.
6. SQL Injection Attacks on Web Applications [Електронний ресурс] / ResearchGate. – Режим доступу: https://www.researchgate.net/publication/320076267_SQL_Injection_Attacks_on_Web_Applications.
7. What Is SQL Injection? [Електронний ресурс] / WhatIsMyIPAddress.com. – Режим доступу: https://whatismyipaddress.com/what-sql-injection.
8. What is Cross-Site Scripting? [Електронний ресурс] / Cloudflare. – Режим доступу: https://www.cloudflare.com/learning/security/threats/cross-site-scripting/.
9. Безпечність та якість веб-додатків: сучасні проблеми та рішення [Електронний ресурс] // Харківський національний економічний університет ім. С. Кузнеця. - 2024. – Режим доступу:

http://repository.hneu.edu.ua/jspui/bitstream/123456789/31827/1/foss-2024-theses.pdf#page=25.

10. Naveen H., Panda B.S., Mohapatra S. Cross-Site Scripting: A Comprehensive Analysis of Attack Techniques and Prevention Strategies [Електронний ресурс] // Research Gate. - 2022. – Режим доступу: https://www.researchgate.net/publication/370131172_Cross-Site_Scripting_A_Comprehensive_Analysis_of_Attack_Techniques_and_Prevention_Strategies.
11. Cross-Site Request Forgery (CSRF) Attack [Електронний ресурс] / Medium.com. – Режим доступу: https://medium.com/@rajeevranjancom/cross-site-request-forgery-csrf-attack-6949edb9e405.
12. What is a Denial of Service (DoS) Attack? [Електронний ресурс] / Wallarm – Режим доступу: https://www.wallarm.com/what/dos-denial-of-service-attack.
13. Distributed Denial-of-Service (DDoS) attack [Електронний ресурс] / ResearchGate. – Режим доступу: https://www.researchgate.net/publication/324562008/figure/fig3/AS:616337260961796@1523957662823/Distributed-Denial-of-Service-DDoS-attack.png.
14. Threats and Vulnerabilities of Cloud Computing: A Review [Електронний ресурс] / ResearchGate. – Режим доступу: https://www.researchgate.net/publication/324562008_Threats_and_Vulnerabilities_of_Cloud_Computing_A_Review.
15. CSRF Tutorial [Електронний ресурс] / University of Memphis. – Режим доступу: https://www.cs.memphis.edu/~kanyang/COMP4420/reading/csrf.pdf.
16. Dynamic Analysis for Security Testing of WEB Based Applications Using Agent Technology [Електронний ресурс] / ResearchGate. – Режим доступу:

https://www.researchgate.net/publication/304571521_Dynamic_Analysis_for_Security_Testing_of_WEB_Based_Applications_Using_Agent_Technology

17. Річний звіт Державного центру кіберзахисту України за 2022 рік [Електронний ресурс]. – Режим доступу: https://cert.gov.ua/files/pdf/SOC_Annual_Report_2022.pdf.
18. The Impact of Manual and Automatic Testing on Software Testing Efficiency and Effectiveness [Електронний ресурс] / ResearchGate. – Режим доступу: https://www.researchgate.net/publication/370526552_THE_IMPACT_OF_MANUAL_AND_AUTOMATIC_TESTING_ON_SOFTWARE_TESTING_EFFICIENCY_AND_EFFECTIVENESS.
19. Основи тестування програмних систем [Електронний ресурс] / КПІ ім. Ігоря Сікорського. – Режим доступу: https://ela.kpi.ua/server/api/core/bitstreams/c40fe58a-aec7-4b80-b052-4a4a38b4272e/content.
20. Manual Testing vs Automation Testing: Pros and Cons [Електронний ресурс] / SIGMA Software University. – Режим доступу: https://university.sigma.software/manual-testing-vs-automation-testing/.
21. 20 Best Open Source Testing Tools in 2023 [Електронний ресурс] / Guru99. – Режим доступу: https://www.guru99.com/uk/best-open-source-testing-tools.html.
22. 23 Website Security Quick Check Services [Електронний ресурс] / H-X-Technology. – Режим доступу: https://www.h-x.technology/ua/blog-ua/23-website-security-quick-check-services-ua.
23. Мельник Г.М. Огляд засобів для тестування веб-додатків [Електронний ресурс] // Вінницький національний технічний університет. - 2022. – Режим доступу: https://ir.lib.vntu.edu.ua/bitstream/handle/123456789/39835/12045.pdf?sequence=3&isAllowed=y.
24. Як новачку перевірити веб-доступність за допомогою WAVE? [Електронний ресурс] / TestMatick. – Режим доступу:

https://testmatick.com/ru/kak-novichku-proverit-veb-dostupnost-s-pomoshhyu-wave/.

25. 6 Types of Vulnerability [Електронний ресурс]. – Режим доступу: https://www.strikegraph.com/blog/6-types-of-vulnerability-scanning.
26. Що таке навантажувальне тестування? [Електронний ресурс]. – Режим доступу: https://sparsim.org/uk/navantazhuvalne-testuvannia-shcho-tse/.
27. Стрес-тестування в тестуванні програмного забезпечення [Електронний ресурс]. – Режим доступу: https://www.zaptest.com/uk/стрес-тестування-в-тестуванні-програ.
28. Web Content Accessibility Guidelines (WCAG) 2.1 [Електронний ресурс]. – Режим доступу: https://www.w3.org/TR/WCAG21/.
29. OpenVAS [Електронний ресурс]. – Режим доступу: https://www.openvas.org/.
30. Port Scanning Techniques [Електронний ресурс]. – Режим доступу: https://nmap.org/book/man-port-scanning-techniques.html.
31. Обнаружение SQL-инъекций в веб-приложениях [Електронний ресурс]. – Режим доступу: https://habr.com/ru/companies/first/articles/709586/.
32. SSL Check [Електронний ресурс]. – Режим доступу: https://hackertarget.com/ssl-check/.
33. WAVE Evaluation Tool [Електронний ресурс]. – Режим доступу: https://chromewebstore.google.com/detail/wave-evaluation-tool/jbbplnpkjmmeebjpijfedlgcdilocofh?pli=.
34. HostedScan [Електронний ресурс]. – Режим доступу: https://hostedscan.com/.
35. Харківський національний університет імені В. Н. Каразіна [Електронний ресурс]. – Режим доступу: http://46.4.23.20/.
36. Apache JMeter [Електронний ресурс]. – Режим доступу: https://jmeter.apache.org/.
37. Load Balancer [Електронний ресурс]. – Режим доступу: https://fotbo.com.ua/load-balancer.

38. Тестування безпеки веб-додатків [Електронний ресурс] // Національний університет "Києво-Могилянська академія". - 2023. – Режим доступу: https://ekmair.ukma.edu.ua/server/api/core/bitstreams/bb04abbf-7854-4663-a3c7-9ac6967dac70/content.
39. Shinde, P.S., Ardhapurkar, S.B. «Cyber security analysis using vulnerability assessment and penetration testing», World Conference on Futuristic Trends in Research and Innovation for Social Welfare, 2016.
40. Павленко К.С. Розробка методу тестування веб-додатків на основі аналізу потоків даних [Електронний ресурс] // Харківський національний економічний університет ім. С. Кузнеця. - 2019. – Режим доступу: http://repository.hneu.edu.ua/bitstream/123456789/27852/1/1_Павленко_Катерина_Сергіївна_122_Комп_ютерні_науки.pdf.
41. BitLy URL Shortener [Електронний ресурс]. – Режим доступу: https://bitly.com/.
42. Test Planning Process [Електронний ресурс]. – Режим доступу: https://app.bitly.com/Bo5ccpT1qhl/home / https://www.guru99.com/test-planning.html.

ДОДАТОК А

Огляд всіх виявлених вразливостей веб-сканером «HostedScan Security» та рекомендації щодо їх вирішення розглянемо у таб. А.1:

Таблиця А.1 – Огляд всіх виявлених вразливостей:

Інструмент	Вразливість	Опис	Рішення
OpenVAS	TCP Timestamps Information Disclosure	Віддалений хост реалізує мітки часу TCP і тому дозволяє обчислювати час безвідмовної роботи. Побічним ефектом цієї функції є те, що час роботи віддаленого хоста іноді можна обчислити.	Щоб вимкнути мітки часу TCP в linux, необхідно додати рядок 'net.ipv4.tcp_timestamps.
	Weak MAC Algorithm(s) Supported (SSH)	Віддалений SSH-сервер налаштовано на дозвіл/підтримку слабких MAC-алгоритмів.	Вимкнути слабкий алгоритм MAC, про який повідомляється.
	Weak MAC Algorithm(s) Supported (SSH)	Віддалений SSH-сервер налаштовано на дозвіл/підтримку слабких MAC-алгоритмів.	Вимкнути слабкий алгоритм MAC, про який повідомляється.
	ICMP Timestamp Reply Information Disclosure	Віддалений хост відповів на запит ICMP-мітки часу. Відповідь про мітку часу - це ICMP-повідомлення, яке є відповіддю на повідомлення про мітку часу. Ця інформація теоретично може бути використана для використання слабких генераторів випадкових чисел на основі часу в інших службах.	Можливі різні способи усунення проблеми: - Повністю вимкнути підтримку мітки часу ICMP на віддаленому хості. -Захистити віддалений хост брандмауером і заблокувати ICMP-пакети, що проходять через брандмауер в обох напрямках.

Продовження таб. А.1.

OWASP ZAP	Cross-Domain Misconfiguration	Завантаження даних веб-браузера може бути можливим через неправильну конфігурацію CORS на веб-сервері	Переконайтеся, що конфіденційні дані не доступні без автентифікації. Налаштуйте HTTP-заголовок «Access-Control-Allow-Origin» на більш обмежений набір доменів, щоб дозволити веб-браузеру застосовувати політику однакового походження у більш обмежувальний спосіб.
	Missing Anti-clickjacking Header	Відповідь не містить ані Content-Security-Policy з директивою 'frame-ancestors', ані X-Frame-Options для захисту від ClickJacking атак.	Необхідно переконатись, що один з них встановлений на всіх веб-сторінках сторінках, які повертаються сайтом/додатком. Крім того, слід розглянути можливість застосування політики безпеки вмісту Content Security Policy директиву «предки фреймів».

Продовження таб. А.1.

	Content Security Policy (CSP) Header Not Set	Політика безпеки контенту – це додатковий рівень безпеки, який допомагає виявляти та нейтралізувати певні типи атак CSP надає набір стандартних HTTP-заголовків, які дозволяють власникам веб-сайтів декларувати схвалені джерела контенту, які браузери можуть завантажувати на цю сторінку – охоплені типи: JavaScript, CSS, фрейми HTML, шрифти, зображення та вбудовані об'єкти такі як Java-аплети, ActiveX, аудіо- та відеофайли.	Переконатись, що веб-сервер, сервер додатків, балансувальник навантаження тощо налаштовано на встановлення заголовка Content-Security-Policy.
	X-Content-Type-Options Header Missing	У заголовку Anti-MIME-Sniffing X-Content-Type-Options не було встановлено значення 'nosniff'. Це дозволяє старим версіям Internet Explorer і Chrome виконувати MIME-сканування тіла відповіді, що потенційно може призвести до інтерпретації та відображення тіла відповіді як типу контенту, відмінного від заявленого. Поточна та застарілі версії Firefox використовуватимуть оголошений тип вмісту замість того, щоб виконувати MIME-розпізнавання.	Переконайтись, що програма/веб-сервер правильно встановив заголовок Content-Type, а також встановив заголовок X-Content-Type-Options значення «nosniff» для всіх веб-сторінок.

Продовження таб. А.1.

	Server Leaks Information via "X-Powered-By" HTTP Response Header Field(s)	Веб-сервер/сервер додатків витікає інформацію через один або кілька заголовків HTTP-відповідей «X-Powered-By». Доступ до такої інформації може полегшити зловмисникам виявлення інших фреймворків/ компонентів, на які покладається веб-додаток.	Переконайтись, що веб-сервер, сервер додатків, балансувальник навантаження тощо налаштовані на придушення заголовків «X-Powered-By».
	Server Leaks Version Information via "Server" HTTP Response Header Field	Веб-сервер/сервер додатків витікає інформацію про версію через заголовок HTTP-відповіді «Server». Доступ до такої інформації може полегшити зловмисникам виявлення інших вразливостей вашого веб-сервера/додатків.	Переконайтись, що ваш веб-сервер, сервер додатків, балансувальник навантаження тощо налаштовані на придушення заголовка «Server».
Nmap	Open TCP Port: 2244	Несподівано відкритий порт може надати ненавмисний доступ до програм, даних і приватних мереж. Відкриті порти також можуть бути небезпечними коли очікувані сервіси застаріли і використовуються через вразливості в системі безпеки.	Важливо перевірити, чи порт 2244 є дійсно необхідним для роботи системи. Якщо він не використовується, рекомендується закрити його для зменшення потенційних вразливостей. Якщо цей порт дійсно використовується необхідно переконатись, що ПЗ, яке використовує цей порт, має всі необхідні оновлення для усунення вразливостей.

Продовження таб. А.1.

	Open TCP Port: 2236	Відкритий порт може бути очікуваною конфігурацією. Несподівано відкритий порт може надати ненавмисний доступ до програм, даних і приватних мереж. Відкриті порти також можуть бути небезпечними коли очікувані сервіси застаріли і використовуються через вразливості в системі безпеки.	Важливо перевірити, чи порт 2244 є дійсно необхідним для роботи системи. Якщо він не використовується, рекомендується закрити його для зменшення потенційних вразливостей. Якщо цей порт дійсно використовується необхідно переконатись, що ПЗ, яке використовує цей порт, має всі необхідні оновлення для усунення вразливостей.
	Open TCP Port: 34568	Відкритий порт може бути очікуваною конфігурацією. Несподівано відкритий порт може надати ненавмисний доступ до програм, даних і приватних мереж. Відкриті порти також можуть бути небезпечними коли очікувані сервіси застаріли і використовуються через вразливості в системі безпеки.	Важливо перевірити, чи порт 2244 є дійсно необхідним для роботи системи. Якщо він не використовується, рекомендується закрити його для зменшення потенційних вразливостей. Якщо цей порт дійсно використовується необхідно переконатись, що ПЗ, яке використовує цей порт, має всі необхідні оновлення для усунення вразливостей. слід застосовувати заходи безпеки.

Продовження таб. А.1.

	Open TCP Port: 80	Відкритий порт може бути очікуваною конфігурацією. Несподівано відкритий порт може надати ненавмисний доступ до програм, даних і приватних мереж. Відкриті порти також можуть бути небезпечними коли очікувані сервіси застаріли і використовуються через вразливості в системі безпеки.	Важливо перевірити, чи порт 2244 є дійсно необхідним для роботи системи. Якщо він не використовується, рекомендується закрити його для зменшення потенційних вразливостей. Якщо цей порт дійсно використовується необхідно переконатись, що ПЗ, яке використовує цей порт, має всі необхідні оновлення для усунення вразливостей. Також слід застосовувати заходи безпеки (брандмауери, VPN, багатофакторна аутентифікація та регулярне оновлення системи).

ДОДАТОК Б

Результати View Results Tree та View Results in Table для 10 користувачів (рис. Б.1 – Б.2):

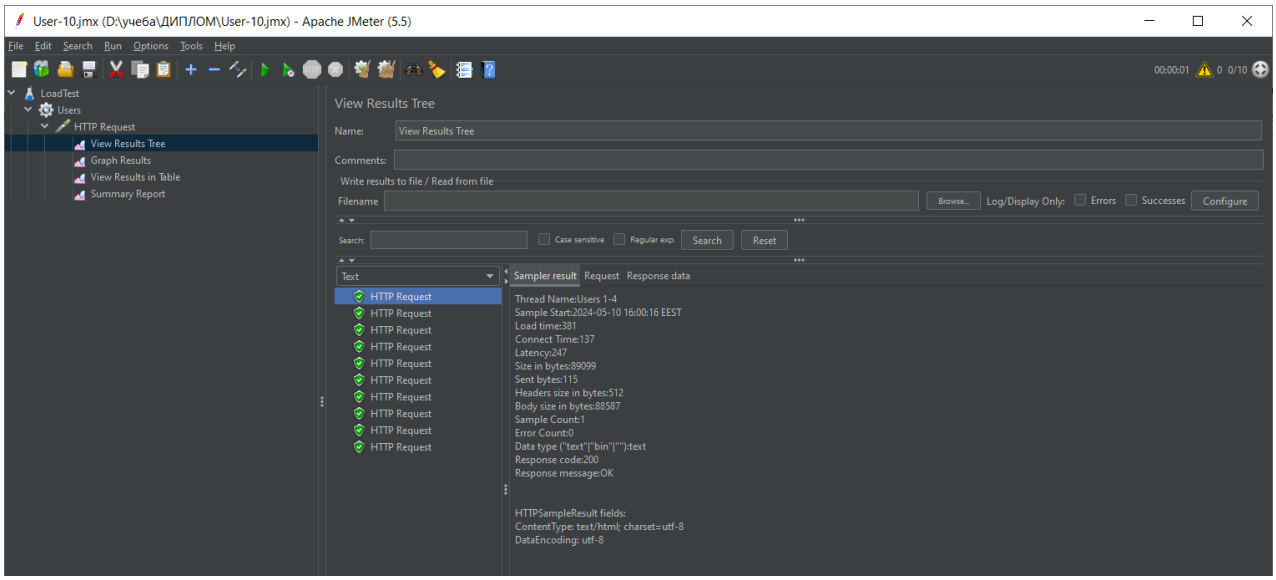


Рисунок Б.1 – View Results Tree

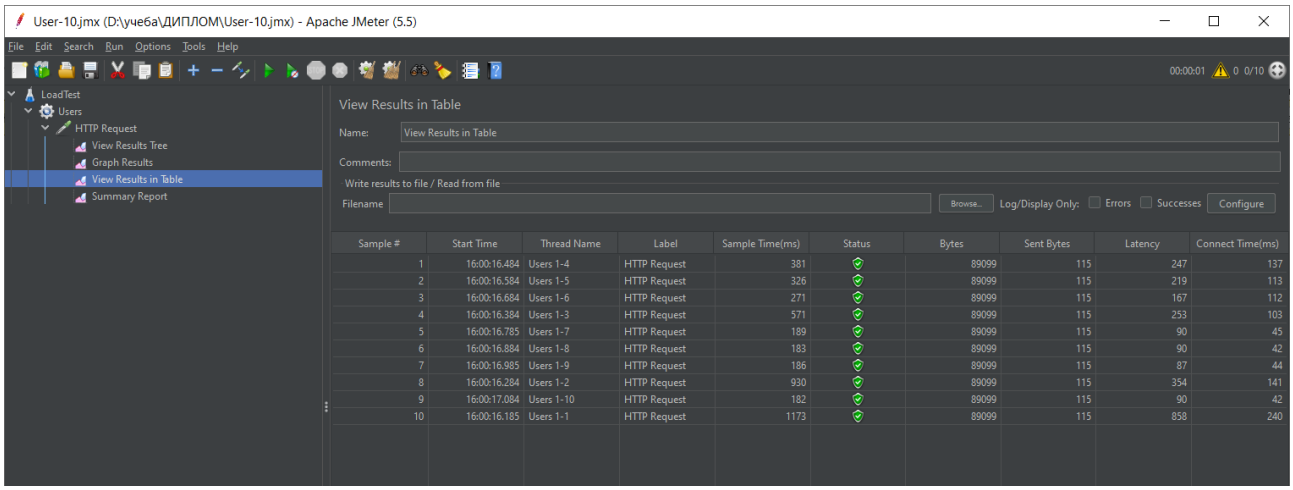


Рисунок Б.2 – View Results in Table

Результати View Results Tree та View Results in Table для 100 користувачів (рис. Б.3 – Б.4):

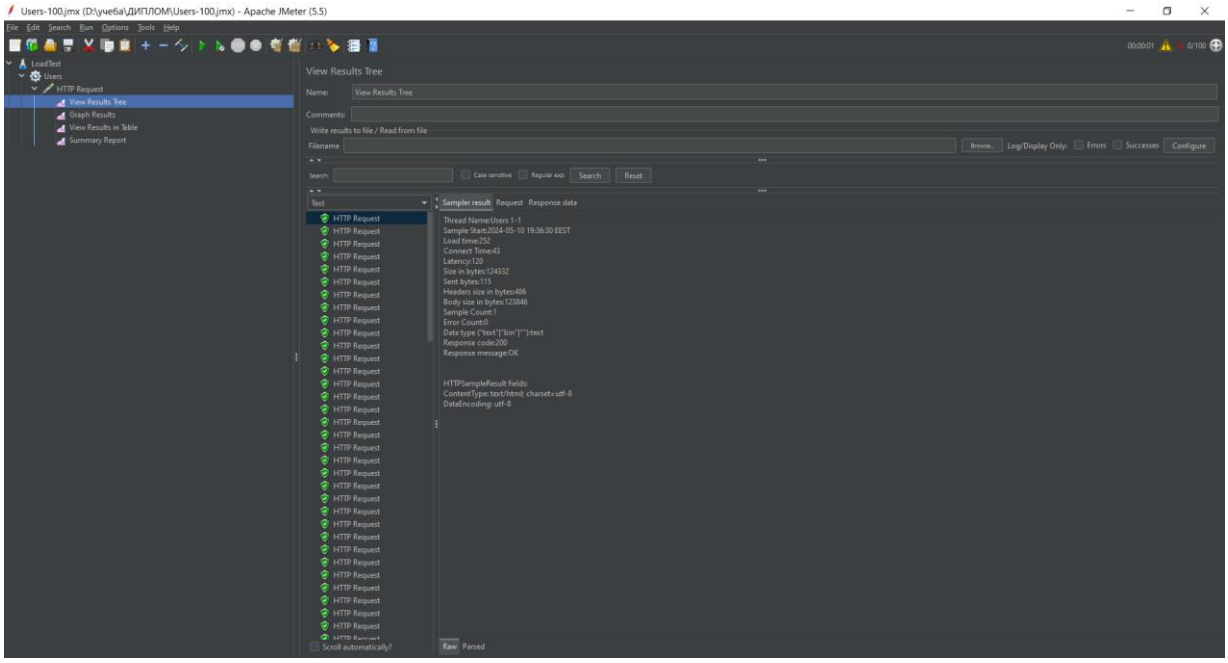


Рисунок Б.3 – View Results Tree

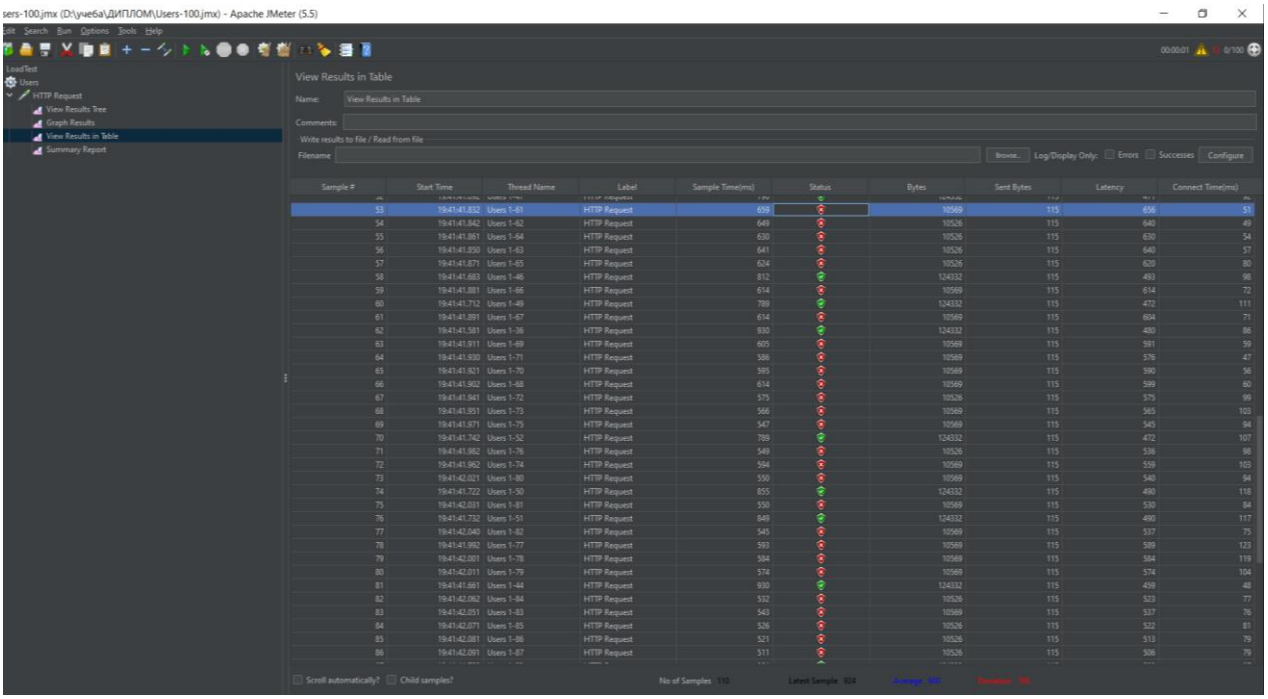


Рисунок Б.4 – View Results in Table

Результати View Results Tree та View Results in Table для 1000 користувачів (рис. Б.5 – Б.6):

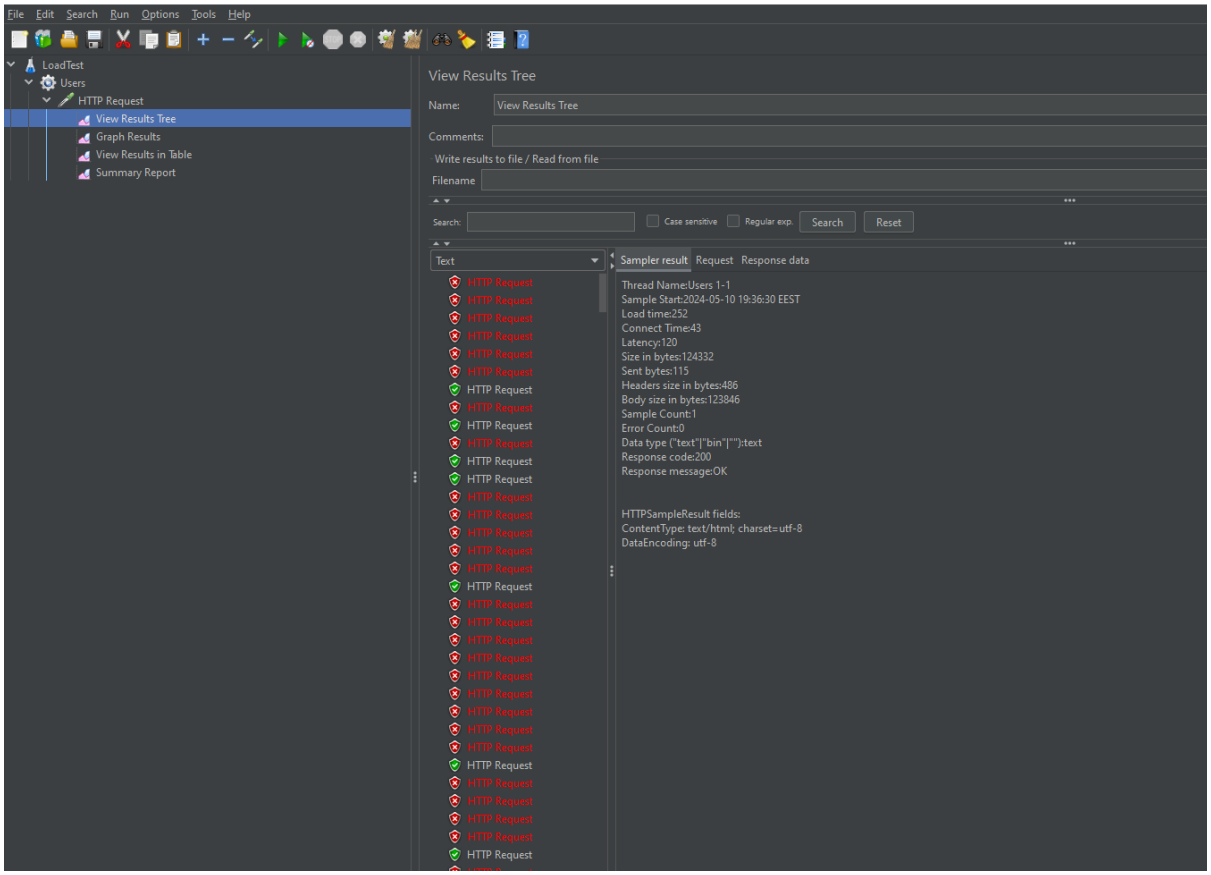


Рисунок Б.5 – View Results Tree

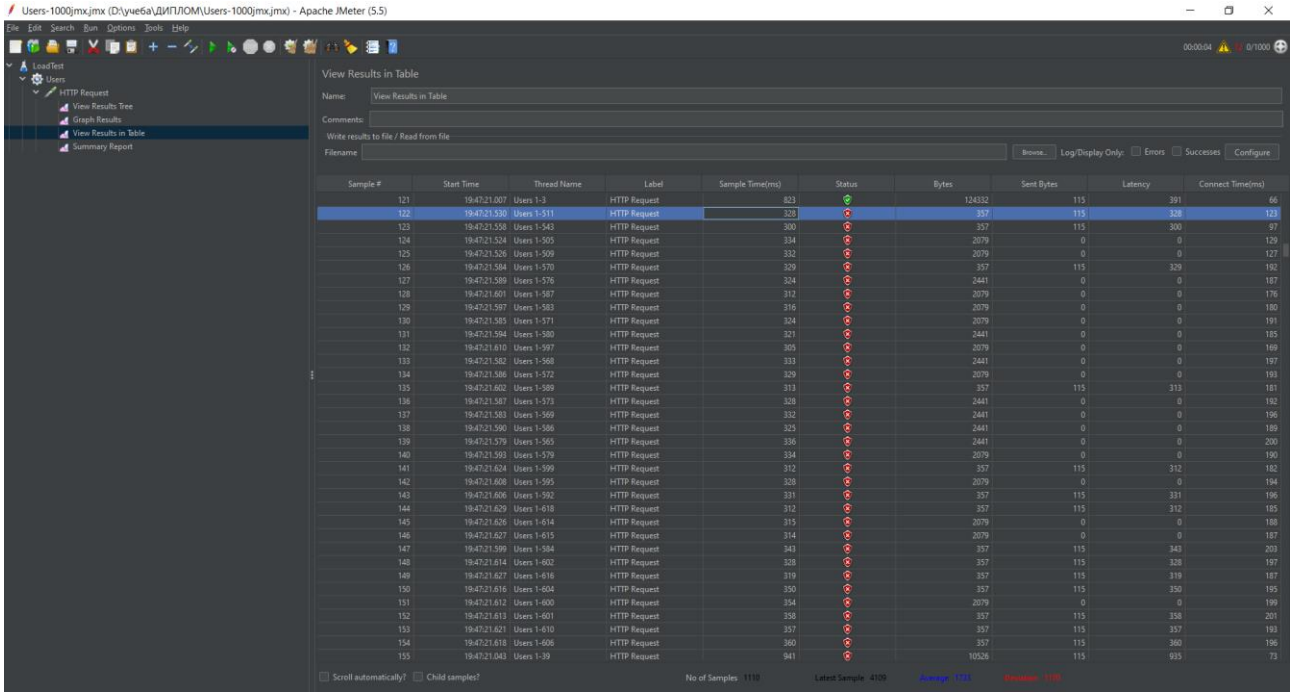
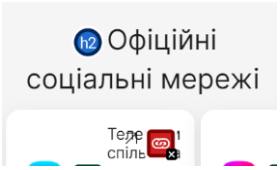
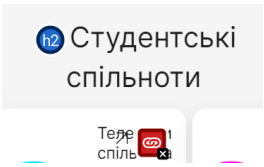


Рисунок Б.6 – View Results in Table

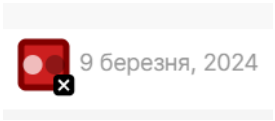
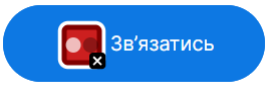
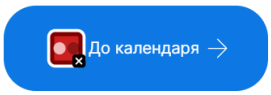
ДОДАТОК В

Огляд всіх виявлених помилок та попереджень доступності за допомогою розширення Google Chrome WAVE та рекомендації щодо їх вирішення розглянемо у таб. В.1:

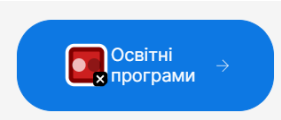
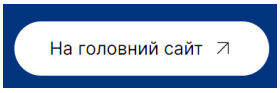
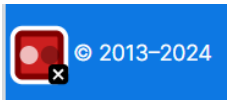
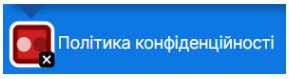
Таблиця В.1 – Огляд помилок та попереджень доступності

Вид	Помилка	Опис	Рішення
Помилки	Empty from label 	Ця помилка означає, що мітка форми присутня, але не містить жодного вмісту. І це дійсно так, якщо натиснути на ці три риски, які видно лише не в повноекранному режимі, то там дійсно пусто.	Щоб виправити цю помилку, необхідно додати відповідний текст до цього спливаючого меню. Підписи не потрібні тільки для елементів управління зображеннями, редагування, скидання, кнопок або прихованих форм.
	Empty link 	Посилання не містить тексту.	Видалити порожнє посилання або додайти до нього текст, що описує функцію та/або мету посилання.
	Empty link 	Посилання не містить тексту.	Видалити порожнє посилання або додайти до нього текст, що описує функцію та/або мету посилання.
	Empty link 	Посилання не містить тексту.	Видалити порожнє посилання або додайти до нього текст, що описує функцію та/або мету посилання.

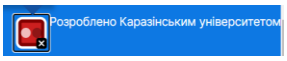
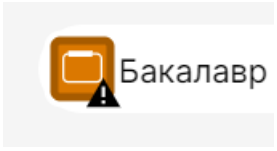
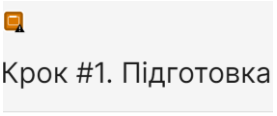
Продовження таб. В.1.

Контрастні помилки	Very low contrast (1.6:1) 	Ця помилка означає, що контраст між текстом і фоном дуже низький, що може заважати сприйняттю написаного людьми з обмеженими можливостями.	Необхідно збільшити контраст між кольором тексту і кольором фону.
	Very low contrast (2.95:1) 	Ця помилка означає, що контраст між текстом і фоном дуже низький, що може заважати сприйняттю написаного людьми з обмеженими можливостями.	Необхідно збільшити контраст між кольором тексту і кольором фону.
	Very low contrast (4.36:1) 	Ця помилка означає, що контраст між текстом і фоном дуже низький, що може заважати сприйняттю написаного людьми з обмеженими можливостями.	Необхідно збільшити контраст між кольором тексту і кольором фону.
	Very low contrast (4.36:1) 	Ця помилка означає, що контраст між текстом і фоном дуже низький, що може заважати сприйняттю написаного людьми з обмеженими можливостями.	Необхідно збільшити контраст між кольором тексту і кольором фону.

Продовження таб. В.1.

	Very low contrast (4.36:1) 	Ця помилка означає, що контраст між текстом і фоном дуже низький, що може заважати сприйняттю написаного людьми з обмеженими можливостями.	Необхідно збільшити контраст між кольором тексту і кольором фону.
	Very low contrast (2.68:1) 	Ця помилка означає, що контраст між текстом і фоном дуже низький, що може заважати сприйняттю написаного людьми з обмеженими можливостями.	Необхідно збільшити контраст між кольором тексту і кольором фону.
	Very low contrast (4.36:1) 	Ця помилка означає, що контраст між текстом і фоном дуже низький, що може заважати сприйняттю написаного людьми з обмеженими можливостями.	Необхідно збільшити контраст між кольором тексту і кольором фону.
	Very low contrast (4.36:1) 	Ця помилка означає, що контраст між текстом і фоном дуже низький, що може заважати сприйняттю написаного людьми з обмеженими можливостями.	Необхідно збільшити контраст між кольором тексту і кольором фону.

Продовження таб. В.1.

	Very low contrast (4.36:1) 	Ця помилка означає, що контраст між текстом і фоном дуже низький, що може заважати сприйняттю написаного людьми з обмеженими можливостями.	Необхідно збільшити контраст між кольором тексту і кольором фону.
Попередження	Missing fieldset 	Група прапорців або перемикачів не вкладається в набір полів.	Визначити, чи має група прапорців або перемикачів текст, який пояснює призначення прапорців або чи потребує він цього. Якщо так, позначити групу в наборі полів і додати опис групи до елемента легенди.
	Missing fieldset 	Група прапорців або перемикачів не вкладається в набір полів.	Визначити, чи має група прапорців або перемикачів текст, який пояснює призначення прапорців або чи потребує він цього.
	Link to PDF document 	Посилання на документ у форматі PDF присутнє.	Переконатись, що PDF-документ доступний за замовчуванням.Крім того, повідомити користувачеві, що за посиланням відкриється документ у форматі PDF.
	Дуже маленький текст 	Текст дуже маленький	Збільште текст до більш читабельного розміру (більше 10 пікселів).

Продовження таб. В.1.

	Присутній відео або аудіо контент 	Це означає, що на сайті присутні відео або аудіо елемент.	Аудіоконтент повинен бути представлений у текстовому форматі, щоб бути повністю доступним для глухих і слабочуючих користувачів. Відеоконтент зі звуком повинен мати синхронізовані субтитри та транскрипт.