МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ХАРЧОВИХ ТЕХНОЛОГІЙ

Інститут (факультет) Автоматизації і комп'ютерних систем
Кафедра інформаційних технологій, штучного інтелекту і кібербезпеки

«До захисту в ЕК»	**«До захисту допущено»**
Директор інституту (декан факультету)	Завідувач кафедри
_____ Андрій ФОРСЮК	_____ Сергій ГРИБКОВ
(підпис) (прізвище та ініціали)	(підпис) (прізвище та ініціали)
«02» червня 2025 р.	«02» червня 2025 р.

КВАЛІФІКАЦІЙНА РОБОТА
НА ЗДОБУТТЯ ОСВІТНЬОГО СТУПЕНЯ БАКАЛАВРА

зі спеціальності 122 комп'ютерні науки
(код і назва спеціальності)

освітньо-професійної програми : Комп'ютерні науки

на тему: Розробка модуля сканування інформаційних систем на вразливості

Виконав: студент 4 курсу, групи 4ск _Скворцов Максим Олександрович
(прізвище та ініціали)

Керівник Мошенський Андрій Олександрович _____
(прізвище та ініціали) (підпис)

Консультанти _____ _____
(прізвище та ініціали) (підпис)

_____ _____
(прізвище та ініціали) (підпис)

_____ _____
(прізвище та ініціали) (підпис)

Рецензент _____ _____
(прізвище та ініціали) (підпис)

Я як здобувач(ка) Національного університету харчових технологій розумію і підтримую політику університету з академічної доброчесності. Я не надавав(-ла) і не одержував(-ла) недозволеної допомоги під час підготовки цієї роботи. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело

Здобувач_____
(підпис)

Київ — 2025р.

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ХАРЧОВИХ ТЕХНОЛОГІЙ Інститут (факультет) Автоматизації і комп'ютерних систем

Кафедра Інформаційних технологій, штучного інтелекту і кібербезпеки
Освітній ступінь бакалавр
Спеціальність 122 комп'ютерні науки
(код і назва)
Освітньо-професійна програма __122 Комп'ютерні науки_____
_____
(назва)

ЗАТВЕРДЖУЮ
Завідувач кафедри інформаційних технологій, штучного інтелекту і кібербезпеки _____ Сергій ГРИБКОВ

_____
« 28 » __квітня_____ 2025 року

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ ЗДОБУВАЧА

_____

1. Тема роботи :Розробка модуля сканування інформаційних систем на вразливості
_____
_____,
керівник роботи к.т.н.Мошенський Андрій Олександрович_____,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом закладу вищої освіти від «28» __квітня__ 2025 р. № _254-кс_
2. Строк подання здобувачем роботи: _30.05.2025_____
3. Вихідні дані до роботи: _____Структура підприємства, доступ до внутрішньої документації, інформація про відділи
_____
_____
4. Зміст пояснювальної записки (перелік питань, які потрібно розробити):
1. Загальна характеристика підприємства, 2. Технічне завдання, 3. Розробка модуля, Висновки, Список джерел інформації, додатки

5. Перелік графічного матеріалу:
_Схема підприємства, схема підрозділу, схема моделі бази даних, структура модуля.

6. Консультанти розділів роботи:

Розділ	**Прізвище, ініціали та посада консультанта**	**Підпис, дата**	
		Завдання видав	**Завдання прийняв**
1	Мошенський А.О. к.т.н.	28.04.2025	29.05.2025
1	Мошенський А.О. к.т.н.	28.04.2025	29.05.2025
1	Мошенський А.О. к.т.н.	28.04.2025	29.05.2025

7. Дата видачі завдання: _____28 квітня 2025 року_____

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів виконання кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Дослідження предметної області	29.04.2025	Виконано
2	Розробка функціональної моделі	30.04.2025	Виконано
3	Розробка концептуальної моделі	01.05.2025	Виконано
4	Розробка технічного завдання	05.05.2025	Виконано
5	Визначення вимог до функцій системи	10.05.2025	Виконано
6	Реалізація задач автоматизації системи	15.05.2025	Виконано
7	Оформлення пояснювальної записки	20.05.2025	Виконано
8	Доопрацювання роботи з урахуванням зауважень керівника	26.05.2025	Виконано
9	Створення презентації	30.05.25	Виконано

Здобувач _____ _____

Керівник роботи _____ _____

АНОТАЦІЯ

Метою кваліфікаційної роботи є розробка спеціалізованого модуля для автоматизованого аналізу інформаційних систем на наявність вразливостей безпеки.

Предметом дослідження є процеси виявлення та класифікації вразливостей в інформаційних системах різного призначення. Реалізація проєкту відбувалась у декілька етапів. На першому етапі було проведено дослідження існуючих методів аналізу вразливостей та визначено оптимальні підходи до їх виявлення. На другому етапі розроблено архітектуру модуля та спроєктовано базу даних для зберігання інформації про вразливості в середовищі MySQL. Третій етап полягав у програмній реалізації серверної частини з використанням мови програмування Python та фреймворку Flask, що забезпечило гнучкість та масштабованість рішення. Четвертий етап включав розробку інтерактивного користувацького інтерфейсу з використанням HTML, CSS та JavaScript, який надає зручний доступ до функціоналу модуля.

Створений модуль дозволяє проводити сканування інформаційних систем, виявляти потенційні вразливості, класифікувати їх за рівнем небезпеки та формувати детальні звіти з рекомендаціями щодо усунення виявлених проблем. Система має інтуїтивно зрозумілий веб-інтерфейс, що спрощує роботу спеціалістів з кібербезпеки та адміністраторів мереж.

Дипломна робота обсягом у 64 сторінок містить 53 сторінок пояснювальної записки, 7 додатків та список із 25 літературних джерел.

Ключові слова: PYTHON, FLASK, MYSQL, HTML, CSS, JAVASCRIPT, ІНФОРМАЦІЙНА БЕЗПЕКА, АНАЛІЗ ВРАЗЛИВОСТЕЙ, ВЕБ-ЗАСТОСУНОК, СКАНУВАННЯ БЕЗПЕКИ, КІБЕРБЕЗПЕКА, ЗАХИСТ ІНФОРМАЦІЇ, ТЕСТУВАННЯ НА ПРОНИКНЕННЯ

SUMMARY

The purpose of the qualification work is to develop a specialized module for automated analysis of information systems for security vulnerabilities.

The subject of the research is the processes of identifying and classifying vulnerabilities in information systems for various purposes. The project was implemented in several stages. At the first stage, we studied existing vulnerability analysis methods and identified the best approaches to detecting them. At the second stage, the module architecture was developed and a database for storing information about vulnerabilities in the MySQL environment was designed. The third stage was the software implementation of the server side using the Python programming language and the Flask framework, which ensured the flexibility and scalability of the solution. The fourth stage included the development of an interactive user interface using HTML, CSS, and JavaScript, which provides convenient access to the module's functionality.

The created module allows scanning information systems, identifying potential vulnerabilities, classifying them by risk level, and generating detailed reports with recommendations for fixing the identified problems. The system has an intuitive web interface that simplifies the work of cybersecurity specialists and network administrators.

The qualification work of 64 pages includes 53 pages of explanatory note, 7 appendices and a list of 25 references.

Keywords: PYTHON, FLASK, MYSQL, HTML, CSS, JAVASCRIPT, INFORMATION SECURITY, VULNERABILITY ANALYSIS,

ЗМІСТ

ВСТУП 7
РОЗДІЛ 1. ДОСЛІДЖЕННЯ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ 9
1.1 Загальна характеристика ТОВ «АМО ФІНТЕХ» 9
1.2. Організаційна структура компані 10
1.3 Аналіз нинішнього стану комп'ютеризації «АМО ФІНТЕХ» 13
1.4 Функціональне моделювання та аналіз існуючих бізнес-процесів 14
1.5 Огляд існуючих рішень для розв'язання існуючих проблем 17
1.6 Розрахунок техніко-економічного обґрунтування впровадження створюваного програмного забезпечення. 18
1.7 Обґрунтування доцільності проєктування й розроблення 19
РОЗДІЛ 2. ТЕХНІЧНЕ ЗАВДАННЯ НА ПРОЄКТУВАННЯ 21
2.1 Загальні положення 21
2.2 Призначення і цілі створення модуля 21
2.3 Вимоги до системи 22
2.4 Вимоги до апаратного забезпечення 23
2.5 Порядок контролю та приймання модуля 24
РОЗДІЛ 3. РОЗРОБКА МОДУЛЯ 25
3.1 Обґрунтування вибору програмно-технічних засобів розроблення програмного продукту 25
3.2 Проєктування та створення бази даних 26
3.3 Реалізація функцій системи 29
3.4 Інструкція користувача 43
3.5 Тестування програмного продукту 44
ВИСНОВКИ 52
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 53
ДОДАТКИ 54

ВСТУП

У сучасному цифровому суспільстві, де інформаційні технології пронизують усі сфери життєдіяльності, проблематика захисту даних та кіберстійкості набуває першочергового значення. Постійне ускладнення архітектури інформаційних систем паралельно з еволюцією методів кібератак формує середовище підвищеного ризику для організацій будь-якого масштабу та галузевої належності.

Актуальність обраної тематики дипломного дослідження зумовлена експоненційним зростанням кількісних та якісних показників несанкціонованих втручань у роботу комп'ютерних мереж та інформаційних систем. Статистичні дані провідних дослідницьких центрів демонструють тривожну тенденцію – часовий проміжок між виявленням нової вразливості та її практичним використанням зловмисниками невпинно скорочується, досягаючи в окремих випадках лише кількох годин. Фінансові та репутаційні наслідки успішних кібератак сягають колосальних масштабів, що підкреслює критичну необхідність розробки ефективних інструментів превентивного характеру.

Метою даної дипломної роботи виступає створення спеціалізованого програмного рішення для комплексного аналізу інформаційних систем на предмет наявності вразливостей безпеки. Реалізація поставленої мети передбачає глибоке дослідження методологічних засад ідентифікації потенційних слабких місць у структурі інформаційних систем, проектування гнучкої архітектури аналітичного модуля, розробку оптимізованої бази даних для документування виявлених вразливостей, імплементацію серверного компонента з використанням сучасного технологічного стеку Python та Flask, а також створення ергономічного користувацького інтерфейсу на основі веб-технологій HTML, CSS та JavaScript.

Об'єктом дослідження визначено багатоаспектні процеси виявлення, класифікації та аналізу вразливостей різнорідних інформаційних систем у контексті забезпечення кібербезпеки.

Предметна область дослідження охоплює прикладні методики, алгоритмічне забезпечення та програмні засоби діагностики інформаційних систем щодо потенційних вразливостей безпеки.

Практична цінність розробленого модуля полягає у створенні інтегрованого інструментарію для суттєвого підвищення захищеності інформаційних активів організацій шляхом автоматизованої ідентифікації вразливостей та формування персоналізованих рекомендацій з їх нейтралізації. Впровадження розробленого рішення дозволяє оптимізувати процеси управління інформаційною безпекою, мінімізувати ризики успішних кібератак та забезпечити належний рівень конфіденційності, цілісності та доступності критично важливої інформації.

Результати проведеного дослідження та розроблений програмний модуль становлять значний інтерес для спеціалістів у галузі інформаційної безпеки, системних архітекторів та адміністраторів, відповідальних за забезпечення кіберстійкості інформаційної інфраструктури різного масштабу та призначення. Запропоноване рішення може бути впроваджене як компонент комплексної системи управління інформаційною безпекою для проведення регулярних безпекових аудитів та превентивного виявлення потенційних загроз.

РОЗДІЛ 1. ДОСЛІДЖЕННЯ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ

1.1. Загальна характеристика ТОВ «АМО ФІНТЕХ»

ТОВ «АМО ФІНТЕХ» займається наданням послуг платіжних сервісів, зокрема це сервіси обробки банківських транзакцій, сервіси електронної комерції та 3D Secure верифікації (Рис. 1.1, 1.2).

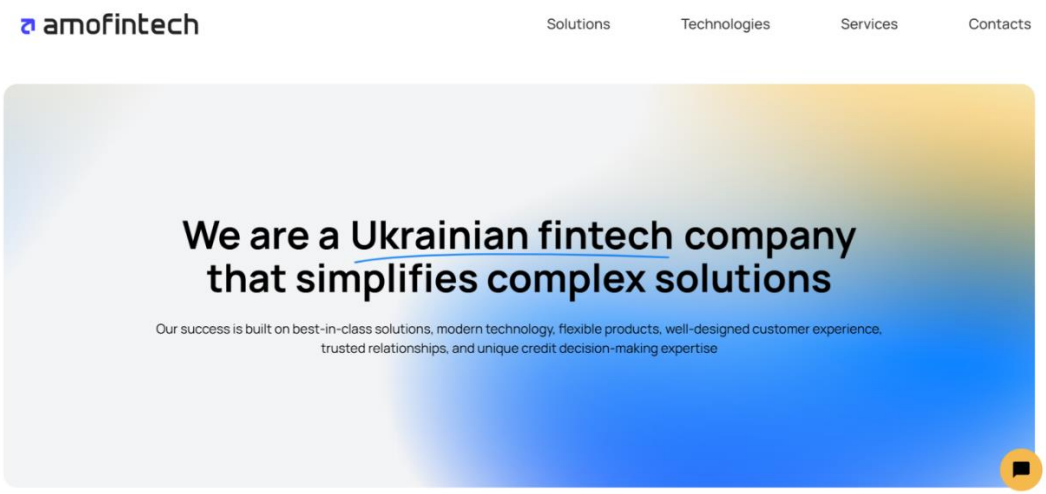

Рисунок 1.1 – Офіційна сторінка ТОВ «АМО ФІНТЕХ»

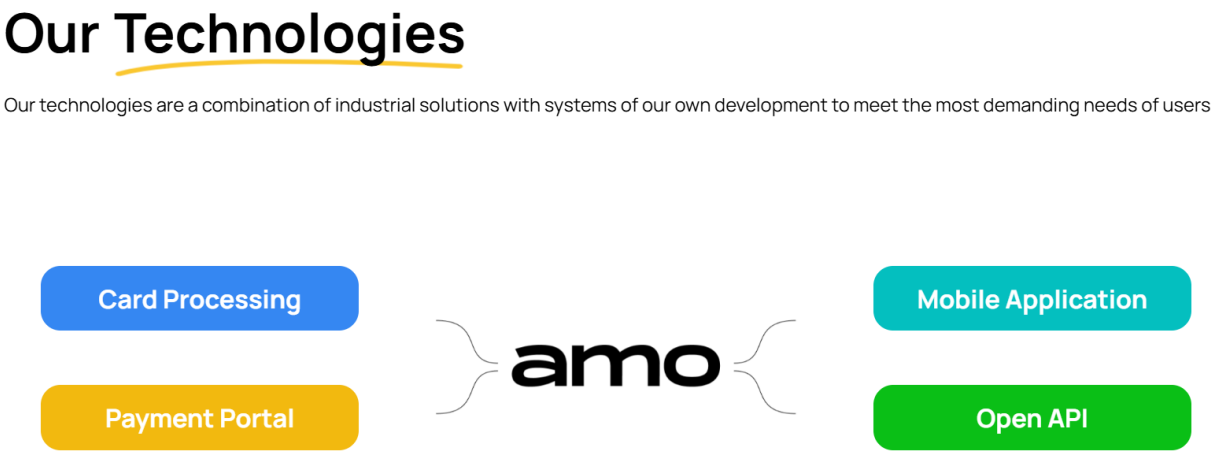

Рисунок 1.2 – Послуги, які надає «АМО ФАНТЕХ»

1.2 Організаційна структура компанії

1.2.1 Загальна схема організаційної структури

Опис, як проводиться управління в організації.

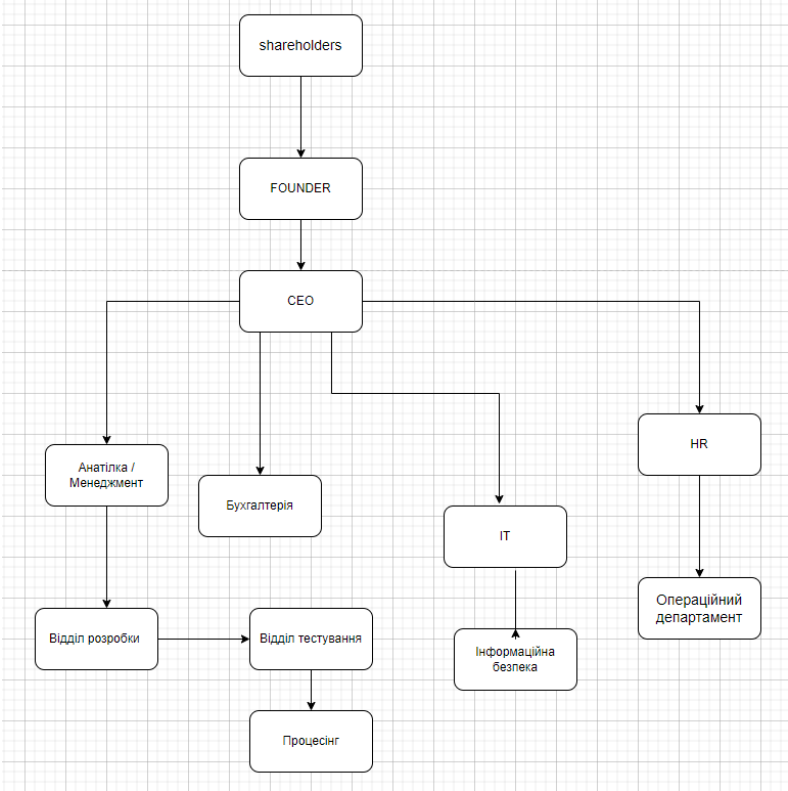

Рисунок 1.3 – Загальна схема організації

Представлена організаційна структура представляє собою ієрархічну модель компанії, що ілюструє вертикальні та горизонтальні зв'язки між різними відділами та ключовими керівними посадами (Рис. 1.3).

Організаційна структура починається на вершині з акціонерів, які надають основну фінансову підтримку і є власниками акцій компанії. Безпосередньо під ними знаходиться засновник, що відіграє вирішальну стратегічну роль у формуванні напрямку розвитку компанії.

Головний виконавчий директор (CEO) займає центральне місце в організаційній структурі, виконуючи роль основного керівника, та стратегічного

лідера. Генеральний директор має прямі зв'язки з багатьма критично важливими відділами.

Звітуючи безпосередньо генеральному директору, аналітичний відділ підтримує тісний зв'язок з відділами розробки та тестування.

На організаційній схемі показано два взаємопов'язані технічні відділи:

- відділ розробки;

- відділ тестування;

Відділ кадрів (HR) :

- займається кадровим забезпеченням, підбором персоналу та відносинами з працівниками;

- пов'язаний з операційним відділом, що передбачає зосередження на управлінні персоналом та операційній ефективності;

IT-відділ:

- містить спеціальний підрозділ інформаційної безпеки;

- забезпечує технологічну інфраструктуру та безпеку для всієї організації;

Бухгалтерія :

- підпорядковується генеральному директору;

- відповідає за фінансовий менеджмент та звітність;

Відділ процесінгу - відповідає за впровадження та підтримку палтіжних сервісів та їх взаємодію з клієнтами та платіжними системами Visa і Mastercard.

1.2.2 Структура підрозділу, який автоматизується

Організаційна структура показує взаємозв'язок спеціалістів, що займаються безпекою процесування карт, управлінням та технічною підтримкою сервісів обробки карткових транзакцій. На схемі показано кілька рівнів взаємодії та контролю в організації (Рис. 1.4).

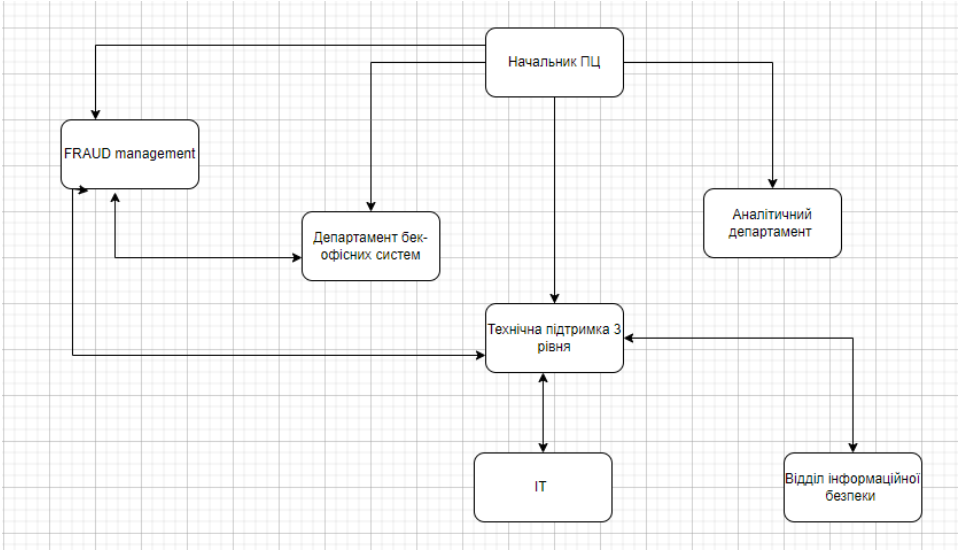

Рисунок. 1.4 – Структура відділу процесингінових систем

Начальник ПЦ :

- займає керівну посаду у відділі.

Безпосередньо керує двома критично важливими відділами:

- аналітичний департамент;

- департамент банківських офісних систем.

Департамент безпеки та управління ризиками:

Управління протидії шахрайству (Fraud Management):

- підтримує двосторонні відносини з Департаментом банківських офісних систем;

- вказує на надійний підхід до виявлення та запобігання шахрайським діям;

Технічна підтримка:

- центральний координаційний пункт для технічних операцій

Безпосередньо пов'язана з:

- департамент інформаційних технологій (IT);

- відділом інформаційної безпеки (Відділ інформаційної безпеки);

1. Аналітичний департамент процесінгу:

- підпорядковується керівнику процесінгового центру;

- відповідає за аналіз бізнес-задач та організації відповідних спеціалістів для вирішення задачі;

2. Департамент бек-офісних систем :

- займається підтримкою взаєморозрахунків з платіжними системами;
- відповідає за актуалізацію курсів валют;
- відповідає за надання бухгалтерських звітів щодо процесування карток.

3. Департамент інформаційних технологій:

- забезпечує фундаментальну технологічну інфраструктуру;
- підтримує операції технічної підтримки.

4. Відділ інформаційної безпеки

- займається питаннями інформаційної безпеки інфраструктури підприємства;
- надає рекомендації для спеціалістів відділу технічної підтримки.

1.3 Аналіз нинішнього стану комп'ютеризації ТОВ «АМО ФІНТЕХ»

У підрозділі що автоматизується зазначені наступні інформаційні системи для управління процесами:

- JIRA – система планування задач, яка дозволяє автоматизувати процеси розробки, тестування, виправлення та затвердження задач з віповідними підрозділами команії.
- Confluence – система, що дозволяє описувати програмні продукти, аналітичні задачі та технічні завдання до виконання. Система відстежувати процес розробки, вдосконалення і виправлення продуктів, що потім будуть реалізовані в продуктивному середовищі.
- Elasticsearch – система логування подій у продуктивному та перед-продуктивному контурах відділу розробки та відділу процесінгу.
- Promehteus – система зробу метрик, що дозволяє прогнозувати роботу сервісів та оперативно реагувати на інциденти у продуктивному середовищі.
- WAZUH – система швидкого реагування на інциденти, пов'язані з інформаціною безпекою та процесуванням карток.

Програмний комплекс у відділі не є оптимізованим у з’язку з тим що жодна з представлених систем моніторингу не є простою у налаштуванні і потребує значного технічного досвіду у налаштуванні.

Ці системи потребують досить багато технічних ресурсів і не завжди оптимізовані для моніторингу мережевих інцидентів пов’язаних з безпекою.

1.4 Функціональне моделювання та аналіз існуючих бізнес-процесів.

1.4.1 Побудова функціональної моделі

Функціональна модель взаємодії передбачає співпрацю з відділами аналітики, інформаційної безпеки і розробки (Рис. 1.5, 1.6, 1.7).

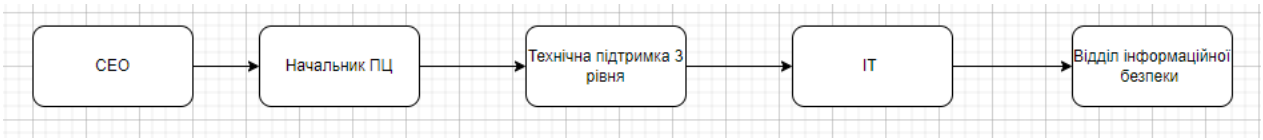

Рисунок 1.5 – Впровадження нового сервісу у продуктивному середовищі

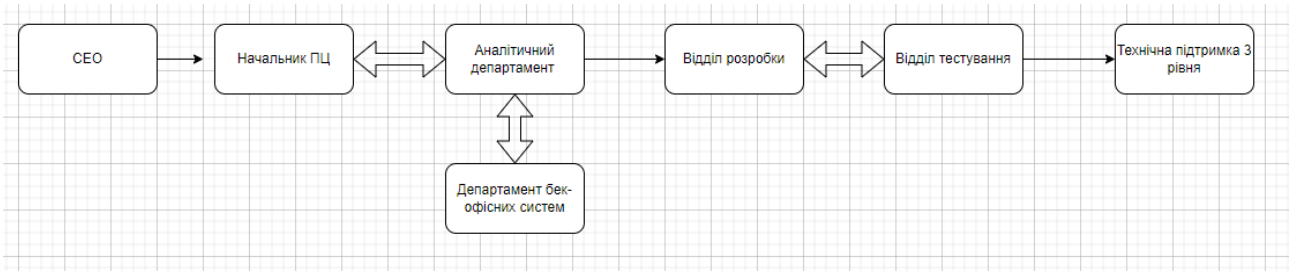

Рисунок 1.6 – Розробка нового сервісу для відділу бек-офісних систем

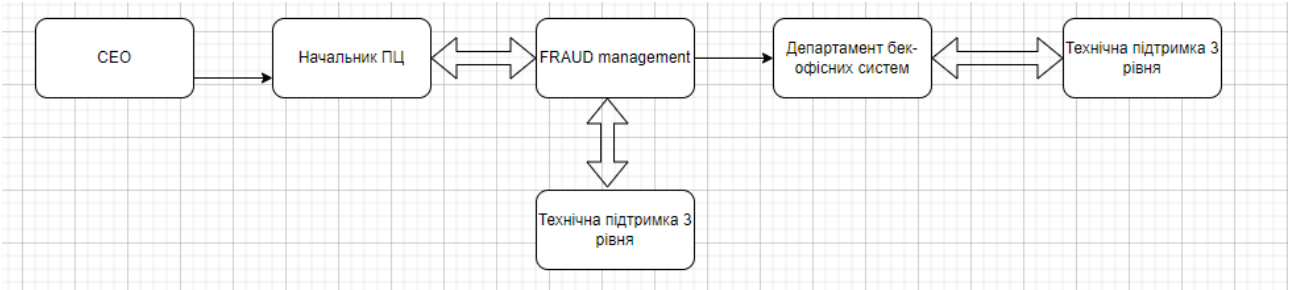

Рисунок 1.7 – Взаємодія між правилами froud management та іншими учсниками підрозділу ПЦ

1.4.2 Виявлені проблеми

Проблемою у даному підрозділі є неефективний моніторинг проблем з мережею, що впливає на статистику безвідмовної працездатності сервісів. У зв’язку з нестабільністю бази даних або мережевого обладнання існуючі системи моніторингу не завжди реагують з оптимальною швидкістю, що впливає на статистику обробки карткових транзакцій, впливає на завантаженість спеціалістів технічної підтримки та моніторингу шахрайства.

У разі численних технічних збоїв спеціалісти технічної підтримки вимушені підтримувати працездатність сервісів і їх відновлення у продуктивному середовищі у будь який час. Критичні проблеми призводять до звернення клієнтів і майбутнього грошого відшкодування.

Дана схема є не тільки не оптимальною але і досить критичною у контексті грошового відшкдування та статистики платіжних сервісів у інфраструктурі Visa і Mastercard.

1.4.3 Пропозиції щодо усунення наявних проблем

Схема роботи системи моніторингу у відділі до вирішення проблеми є не досконалю і має деякі недоліки, такі як:

- Не вчасне реагування на інциденти;
- Погана підготовка до аудиту;
- Не налагоджена робота співробітників (Рис. 1.8).

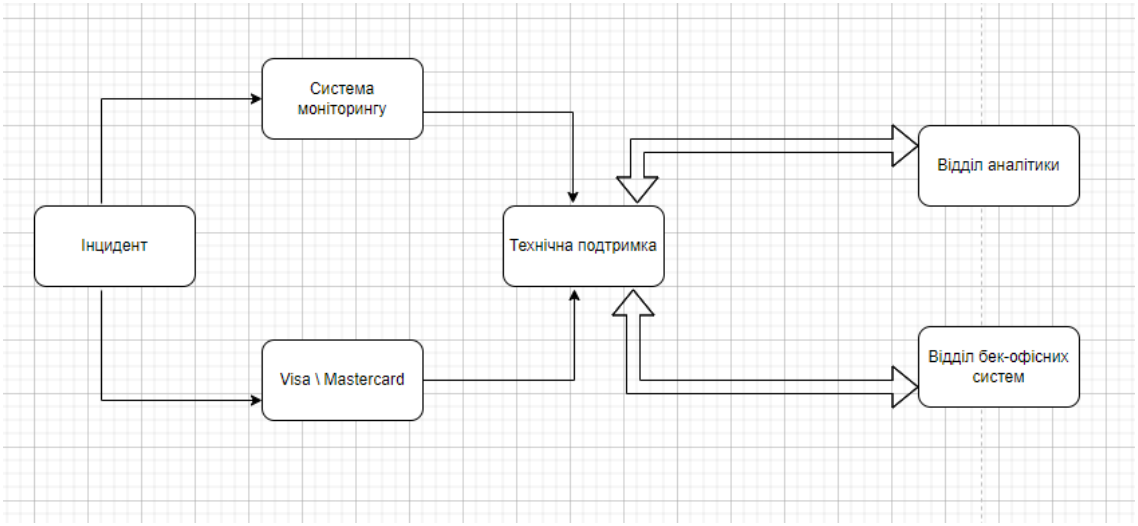

Рисунок 1.8 – Схема AS IS

Схема роботи системи моніторингу у відділі процесінгу після вирішення проблеми повинна посприяти більш високим показникам відмово стійкості та забезпечить більш оптимальну і гнучку підготовку до аудиту інформаційних систем (Рис. 1.9).

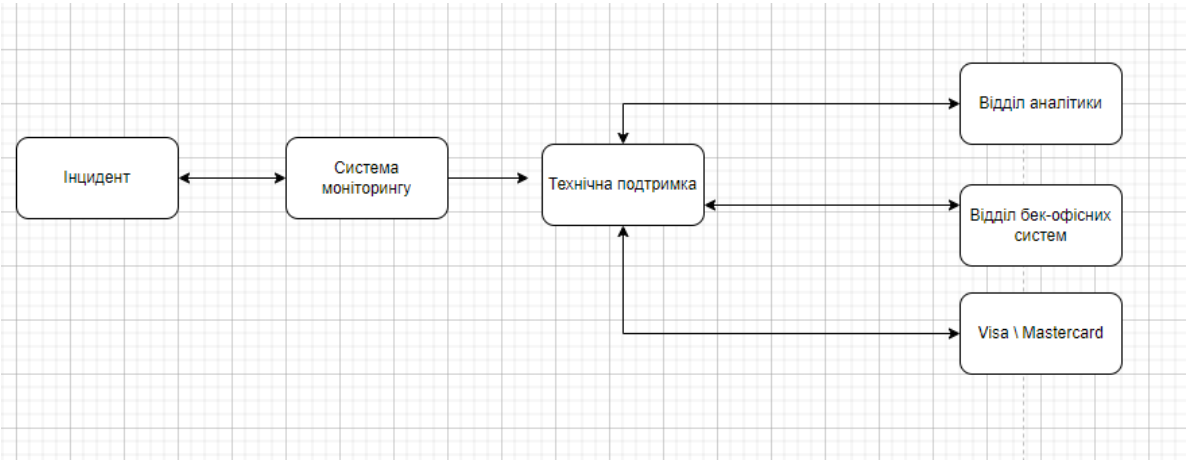

Рисунок 1.9 – Схема TO BE

1.5 Огляд існуючих рішень для розв’язання існуючих проблем

Zabbix:

Opensource рішення для комплексного моніторингу IT-інфраструктури. Підтримує моніторинг мережевих пристроїв, серверів, віртуальних машин та хмарних сервісів.

Nagios:

Забезпечує моніторинг доступності та продуктивності систем. Має розширену систему сповіщень та звітності.

PRTG Network Monitor:

Добре підходить для корпоративного моніторингу. Підтримує моніторинг мережевої інфраструктури, серверів, додатків.

Кожна з цих систем має свої переваги і недоліки (Табл. 1.1).

Таблиця 1.1. Порівняльна таблиця систем

Характеристика	**Zabbix**	**Nagios**	**PRTG Network Monitor**
Тип	Opensource	Opensource	Комерційне рішення
Опис	Комплексний моніторинг IT-інфраструктури	Класичне рішення для моніторингу мережевої інфраструктури	Корпоративне рішення для моніторингу
Об'єкти моніторингу	Мережеві пристрої, сервери, віртуальні машини, хмарні сервіс	Мережева інфраструктура, доступність та продуктивність систем	Мережева інфраструктура, сервери, додатки
Основні переваги	- Безкоштовна базова версія - Висока масштабованість - Підтримка великої кількості плагінів - Зручний веб-інтерфейс	- Стабільність та надійність - Широка підтримка спільноти - Гнучка система налаштування плагінів	- Простота налаштування - Дружній інтерфейс - Широкий набір готових сенсорів - Детальна аналітика
Особливості	Виявлення аномалій та генерування звітів	Розширена система сповіщень та звітності	Простота використання та готові рішення

1.6 Розрахунок -економічного ефекту від впровадження ІС

Для розрахунку техніко-економічного ефекту від впровадження ІС були використані наступні етапи:

1. Визначення витрат на впровадження ІС:

- Прямі витрати (придбання програмного забезпечення, апаратного забезпечення);
- Непрямі витрати (навчання персоналу, тощо)

2. Оцінка вигод від впровадження ІС:

- Підвищення продуктивності праці;
- Поліпшення якості обслуговування клієнтів;

3. Розрахунок техніко-економічного ефекту:

- Визначення терміну окупності;
- Розрахунок чистого приведеного доходу (NPV);
- Розрахунок внутрішньої норми доходу (IRR).

Загальна вартість визначається як сума витрат: $V\Sigma=V1+V2+V3+V4$

Розрахунок V1V_1V1 — Витрати на розробку програмного забезпечення

Формула: $V1=Vоплата+Vпоточні\_витрати$

Оплата праці розробників: $Vоплата = ЗП\ місячна \times М \times Ч$

ЗП місячна = 25000грн — середня зарплата розробника.

М = 3міс — термін розробки.

Ч = 5 — кількість розробників.

$Vоплата = 25000 \times 3 \times 5 = 375000грн$

Поточні витрати:

$Vпоточні\_витрати = З\ ремонт + З\ електроенергія + З\ амортизація$

Ремонт: $Зремонт=6\% \times ЦПК$, ЦПК=40000

$З\ ремонт = 0.06 \times 40000 = 2400грн$

Електроенергія: $Зелектроенергія = РПК \times ТПК \times Цел$

РПК=0.5кВт, ТПК=1892, Цел = 1.86 грн/кВт·годЦ

$Зелектроенергія=0.5 \times 1892 \times 1.86=1760грн$

Амортизація: Замортизація = 5ЦПК = 40000\5 = 8000грн

Vпоточні_витрати = 2400+1760+8000 = 12160грн

V1 = 375000+12160 = 387160грн

Загальна вартість розробки:

VΣ=V1+V2+V3+V4

VΣ=387160+50000+0+10000=447160грн

Розрахунок річного прибутку (Пр)

Припустимо, що впровадження модуля дозволяє:

1. Обробляти більше замовлень, що приносить +150,000 грн/рік.
2. Зменшити кількість помилок, економлячи +100,000 грн/рік.

Пр = 150000+100000 = 250000грн/рік

Розрахунок коефіцієнта економічної ефективності (Кеф)

Кеф=250000\447160 ≈ 0.56

Розрахунок терміну окупності (Ток)

Ток = 1\0.56 ≈ 1.79 року

Висновки

1. Загальна вартість розробки: 447,160 грн.
2. Очікуваний річний прибуток: 250,000 грн.
3. Термін окупності: приблизно 1.8 року.
4. Коефіцієнт економічної ефективності: 0.56.

1.7 Обґрунтування доцільності проєктування й розроблення

Модуль аналізу інформаційних систем на вразливості повинен забезпечити більш раціональну витрату ресурсів компанії та забезпечити відділ процесінгу відповідними звітами щодо інцидентів інформаційної безпеки.

Порівняно з іншими системами швидкого реагування на подібні випадки, модуль, який буде розроблено під час дипломної роботи – буде більш оптимізованим а також потребуватиме значно менше трудовитрат на його

адміністрування і технічну підтримку. Він дає змогу перевіряти інформаційну систему як локально, так і зі світу, тобто глобально.

Порівняно з іншими системами швидкого реагування, модуль аналізу інформаційної системи на вразливості буде мати досить гнучку структуру і не не змушуватиме переплачувати високу вартість за ліцензію.

РОЗДІЛ 2. ТЕХНІЧНЕ ЗАВДАННЯ

2.1 Загальні положення

Модуль аналізу інформаційних систем на вразливості.

1. Результати розробки оформлюються у вигляді:

- Програмного забезпечення на носіях інформації;
- Пояснювальної записки до бакалаврської кваліфікаційної роботи;

2.2 Призначення і цілі створення системи

Модуль призначений для мережевого аналізу інформаційних систем на наявність відкритих не шифрованих сокетів, сокетів, що використовують сертифікати з не надійним шифруванням та низька відмовостійкість до атак типу DOS, DDOS.

Цілі створення системи:

- Забезпечення можливості зовнішнього і внутрішнього сканування інформаційних систем на наявність не безпечних відкритих сокетів та не надійних сервісів рівня http протоколу;

- Формування звітів про виявлені вразливості з рекомендаціями щодо їх усунення;

- Зберігання історії сканувань та відстеження змін у стані безпеки системи

- Надання зручного веб-інтерфейсу для керування процесом сканування та аналізу результатів.

2.3. Вимоги до системи

2.3.1 Вимоги до функцій системи

Основні функції модуля наведено в таблиці 2.1

Таблиця 2.1 - Основні функції системи

№	Назва функції	Вхідні дані	Вихідні дані
1	Робота з хостами (додавання, видалення, редагування)	Ім'я, IP адреса,версія ОС, опис	Додавання інформації в базу даних
2	Сканування мережевих портів	Ввод діапазону портів до сканування	Статус порта, дані про сертифікат
3	Аналіз відмовостійкості	Ввод кількості HTTP запитів на цільову машину	Дані про відпрацьовані запити, коди відповіді
4	Сторінка з графіками	Ввод імені або ір адреси	Дані про графіки з відображенням даних про відмовостійкість
5	Дії з користувачами (додавання, виалення, редагування)	Ввести дані користувача	Запис відповідної інформації в базу даних

2.3.2 Вимоги до графічного інтерфейсу

- Інтуїтивно зрозумілий інтерфейс;
- Адаптивний дизайн для різних пристроїв;
- Інформативна візуалізація результатів сканування;
- Кольорове кодування рівнів критичності вразливостей.

2.3.4 Вимоги до програмного забезпечення

- Мова програмування: Python 3.9+;

- Python Бібліотеки для аналізу вразливостей: socket, requests, ssl, cryptography;

- Фреймворк для веб-інтерфейсу: Flask;

- СУБД : MySQL.

2.3.5 Вимоги до апаратного забезпечення

- Серверна частина:

- Процесор: не менше 2 ядер;

- Оперативна пам'ять: не менше 1 ГБ;

- Дисковий простір: не менше 20 ГБ;

- Мережевий інтерфейс: до 1 Гбіт/с;

Клієнтська частина:

- Сучасний веб-браузер з підтримкою HTML5 та JavaScript;

2.4 Порядок контролю та приймання модуля

- Відповідність функціональним вимогам, зазначеним у ТЗ;

- Відсутність критичних помилок;

- Повнота документації;

- Успішне проходження всіх тестів.

2.5 Календарний план і діаграма Ганта

Було створено таблицю, в якій описано послідовні стадії виконання робіт для реалізації функціоналу (Табл. 2.2).

Таблиця 2.2 – Календарний план

№	Назва етапів виконання кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Дослідження предметної області	29.04.2025	Виконано
2	Розробка функціональної моделі	30.04.2025	Виконано
3	Розробка концептуальної моделі	01.05.2025	Виконано
4	Розробка технічного завдання	05.05.2025	Виконано
5	Визначення вимог до функцій системи	10.05.2025	Виконано
6	Реалізація задач автоматизації системи	15.05.2025	Виконано
7	Оформлення пояснювальної записки	20.05.2025	Виконано
8	Доопрацювання роботи з урахуванням зауважень керівника	26.05.2025	Виконано
9	Створення презентації	30.05.25	Виконано

На основі календарного плану ми будуємо діаграму Ганта (Рис. 2.1).

Рисунок 2.1 – діаграма Ганта

РОЗДІЛ 3. РОЗРОБКА ЕЛЕМЕНТІВ СИСТЕМИ

3.1 Обґрунтування вибору програмно-технічних засобів розроблення програмного продукту

1. Для розробки модуля аналізу інформаційних систем на вразливості було обрано мову програмування python. Ця мова надаж можливість писати код швидко, має багато зручних бібліотек, які стануть у нагоді.

python просто інтегрувати з іншими системами, він не потребує налаштування скадного оточення.

2. Для написання серверної частини було використано саме python flask, у зв’язку з тим, що він має досить гнучкий функціонал, інтуїтивно зрозумілий синтаксис та просту структуру.

3. Python socket –було обрано цю бібліотеку для реалізації методу сканування портів на рівні TCP протоколу. Ця бібліотека є досить гнучкою у розробці, пдтримує багато протоколів транспортного рівня, підтримує багатопоточність і кросплатформенність.

4. Python SSL – було обрано цю бібліотеку для роботи з сертифікатами і визначення даних. Бібліотека SSL є кросплатформенною і зручною у використанні, має зрозумілий синтаксис.

5. Python requests – було обрано цю бібліотеку для реазізації методу перевірки на DDOS атаки. Підтримує багато методів протоколу HTTP, таких як GET, POST, PUT, DELETE та ін. Має зрозумілий синтаксис і кросплатформеність.

6. Javascript – мова, яка є не замінимою у написання функціоналу для роботи зі структурою HTML документу.

3.2 Проектування бази даних

У якості СУБД було вирішено використовувати MySQL. Це рішення зумовлено простотою розгортання такой бази порівняно з такими конкурентами як Oracle чи PostgreSQL (Рис. 3.1).

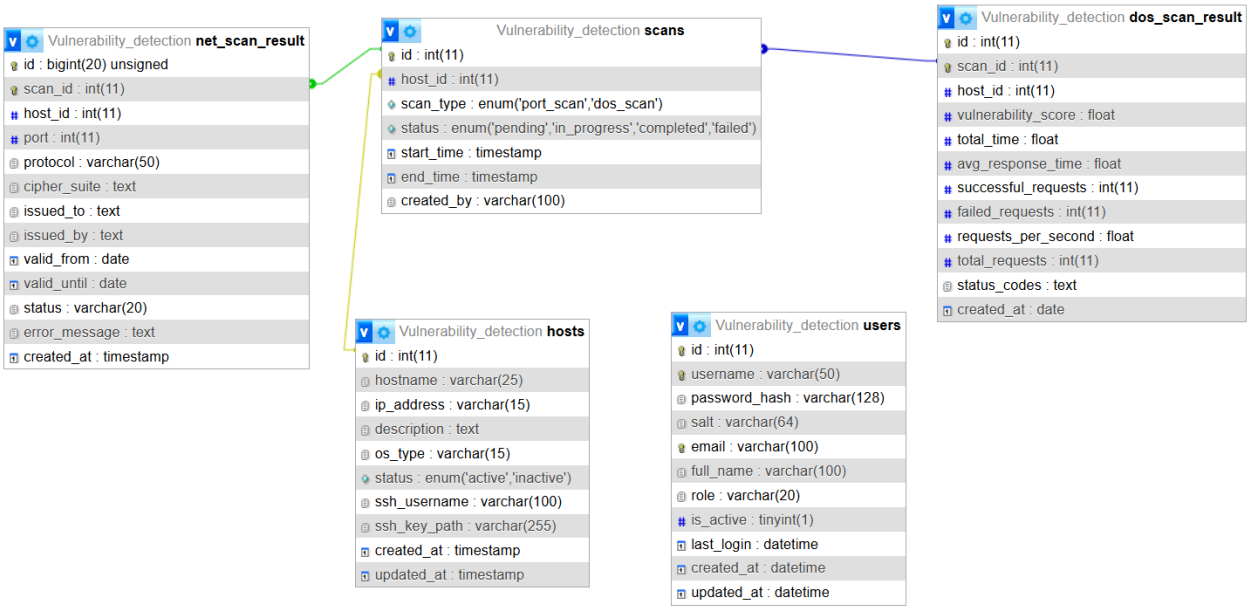

Рисунок 3.1 – Структура бази даних модуля аналізу

Опис бази даних:

1. Таблиця scans – основана таблиця із записами сканувань. Має такі поля:

- Host_id – id цільового хоста, сканування якого було проведено. Має посилання на поле id у таблиці hosts;
- Scan_type – поле, що вказує на тип сканування, яке було проведено. Має фіксовані значення (enum) – port_scan, dos_scan;
- Status – статус сканування в поточний момент часу;
- Created_by, start_time, end_time – часові мітки що відповідають за дату створення запису, час початку і час кінця сканування відповідно.

2. Таблиця net_scan_result – містить дані про мережеві порти та наявні сертифікати :

- Scan_id – посилання, яке пов’язане із колонкою ID в таблиці scans;
- Host_id – ідентифікатор цільової системи, яку було просканово;
- Port – номер мережевого порта, який було скановано;
- Protocol – версія TLS протоколу, за якою створено сертифікат, що знаходиться на порту;

- Cipher_suite – алгоритм шифрування, який було використано при генерації сертифікату;
- Issued_to – власник сертифікату;
- Issued_by – центр сертифікації, що підписував сертифікат;
- Valid_from, valid_to – термін генерації і термін закінчення терміну придатності сертифікату відповідно;
- Status – статус мережевого порта в момент сканування (відкритий, фільтрується фаерволом, ін);
- Error_message – повідомлення про поилку, якшо воно виникло під час сканування;
- Created_at – дата створення запису в базі.

3. Таблиця dos_scan_result – містить результати сканування на атаки типу DOS, DDOS:

- Scan_id – ідентифікатор сканування, яке було проведено;
- Host_id – ідентифікатор цільової системи, що було про скановано;
- Vulnerability_score – попередня оцінка хостової системи, яка рахується на основі HTTP кодів відповіді під час сканування;
- Total_time – сумарна кількість часу, що було витрачено на сканування цільової системи;
- Avg_response_time – середній час відповіді на запит;
- Successful_requests – кількість успішних запитів під час сканування;
- Failed_requests – кількість не успішних запитів під час сканування;
- Request_per_second – кількість запитів в секунду під час сканування;
- Total_requests – сумарна кількість HTTP - запитів, яка була відправлена під час сканування;
- Status_codes – HTTP коди відповіді, які було отримано під час сканування;
- Created_at – дата створення запису в таблиці.

4. Таблиця hosts – зберігає інформацію про цільові системи (хости), ікі можуть бути просканованні:

- id – ідентифікатор запису, посилається на колонку host_id в таблиці scans;
- hostname – DNS імя’я хостової системи;
- ip_address – IP адреса хостової системи;
- description – опис хостової системи;
- os_type – тип операційної системи на хості;
- status – значення enum – active \ inactive – вказує, обслуговується система на разі чи ні, чи може вона підлягати скануванню;
- ssh_user_name, ssh_key_pass – ім’я користувача та шлях до ssh ключа відповідно. Ці дані можуть використовуватись у результаті внутрішніх сканувань. Наразі не використовуються, тому не є обов’язковими. Ці значення можуть бути використані в майбутньому, у разі модернізації модуля аналізу інформаційних систем на вразливості;
- created_at, updated_at – час створення та оновлення запису в базі відповідно.

5. Users – таблиця, що зберігає інформацію про користувачів:

- Username – ім’я користувача в системі, використовується як логін для входу;
- Password_hash – хеш дані пароля користувача. Паролі до облікового запису ніколи не зберігаються в базі даних у відкритому вигляді;
- Salt – так звана «сіль», яку було використано під час хешування паролю користувача;
- Email – адреса електронної погти користувачів;
- Full_name – повне ім’я користувача, може бути використане для звітності;
- Role – роль користувача, наразі не використовується. Може бути використано в мабутньому, в якості додаткового функціоналу - для регулювання прав доступу користувачів;
- Last_login – час останнього успішного входу в систему;

- is_active – поле, що вказує активний користувач чи ні. У разі якшо користувач не активний – від не має доступу до входу в систему;
- Created_At, updated_at – час створення та оновлення запису в базі даних відповідно.

3.3. Реалізація функціоналу модуля

Перед початком роботи необхідно зробити функціональну модель, яка відобразить взаємодію основних компонентів модуля між собою (Рис. 3.2).

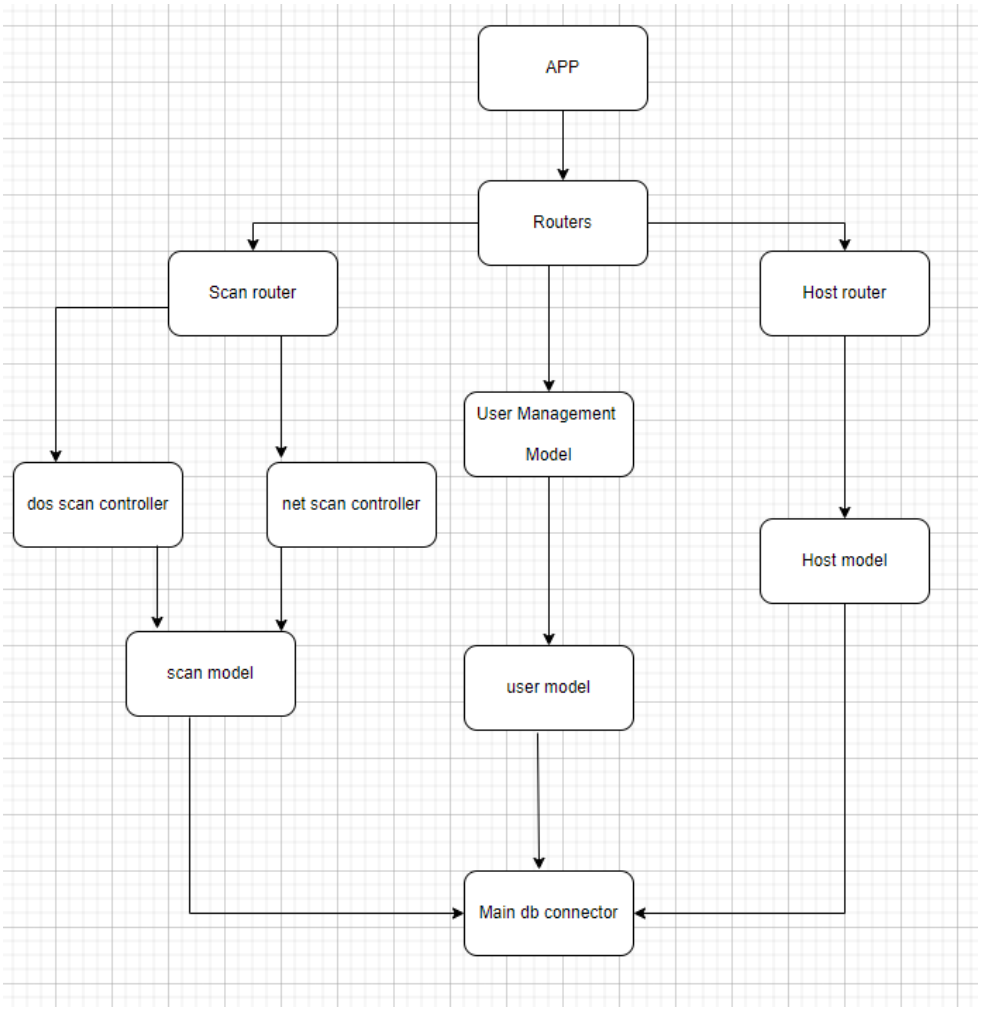

Рисунок 3.2 – Структура модуля аналізу інформаційних систем на вразливості

Опис структури:

1. APP – головна точка входу в додаток, клас, що виконує зв’язок між роутерами, логуванням та запуском основного функціоналу додатку (Рис. 3.3).

```python
class VulnerabilityAnalysisApp:

    def __init__(self, config=None):

        self.app = Flask(__name__)

        if config:
            self.app.config.update(config)
        else:
            self.app.config.update(
                SECRET_KEY=os.environ.get('8uhp7hp977t97', '43tgerg5yh4ht54t43t34tyte
                DEBUG=False
            )

        logging.basicConfig(
            level=logging.INFO,
            format='%(asctime)s - %(name)s - %(levelname)s - %(message)s'
        )
        self.logger = logging.getLogger(__name__)

        self.db = Database()

        self.user_model = UserModel(self.db)
        self.router = Router(self.app, self.user_model, self.logger)
        self.router.setup_routes()

    def run(self, host='localhost', port=5000, debug=False):

        self.logger.info(f"Запуск додатку на {host}:{port} (debug={debug})")
        self.app.run(host=host, port=port, debug=debug)

if __name__ == "__main__":

    app_config = {
        'SECRET_KEY': os.environ.get('f344yer45y6', '55ghg6u5e674hk8orwe90h84o569'),
        'DEBUG': os.environ.get('DEBUG', False)
    }
```

Рисунок 3.3 – Фрагмент коду, який відповідає за запуск веб додатку

В ньому було описано ініціалізацію додатку – self.app = Flask(__name__)

Було визначено основні маршрути дотаку –

self.router = Router(self.app, self.user_model, self.logger)

Було визначено модель для роботи з користувачами –

self.user_model = UserModel(self.db)

2. Routers – клас, що описує правила основних маршрутів, які будуть використовуватись під час роботи веб додатку.

Було написано основний роутер, який відповідає за маршрутизацію головного меню, вхід і вихід користувача до інтерфейсу програми та сторінку з налаштуваннями (Рис. 3.4).

```python
def setup_routes(self):

    self.app.add_url_rule('/login', 'login', self.login, methods=['GET', 'POST'])
    self.app.add_url_rule('/logout', 'logout', self.logout)
    self.app.add_url_rule('/', 'home', login_required(self.home))
    self.app.add_url_rule('/dashboard', 'dashboard', login_required(self.dashboard))
    self.app.add_url_rule('/settings', 'settings', login_required(self.settings))
    self.app.add_url_rule('/settings/update', '/settings/update', login_required(self.update_settings), methods=["POST"])
    self.app.add_url_rule('/settings/users-management', '/settings/users-management', login_required(self.user_management), methods=["POST"])
    self.app.add_url_rule('/user/add', '/user/add', login_required(self.add_user), methods=["POST"])
    self.app.add_url_rule('/user/del', '/user/del', login_required(self.del_user), methods=["POST"])
    self.app.add_url_rule('/user/edit', '/user/edit', login_required(self.edit_user), methods=["POST"])
```

Рисунок 3.4 – Метод, що описує правила обробки основних маршрутів

Кожен з основних маршрутів повинен перевіряти, чи не скінчився час сесії користувача у веб додатку. Це правило є хорошою практикою, з метою підвищення інформаційної безпеки (Рис. 3.5).

```python
def login_required(f):

    @wraps(f)
    def decorated_function(*args, **kwargs):
        with open("cfg/config.yml", 'r') as cfg_f:
            config = yaml.safe_load(cfg_f)

        if 'logged_in' not in session:
            flash('Будь ласка, увійдіть до системи', 'warning')
            return redirect(url_for('login'))

        now = time.time()
        if now - float(session.get('last_active', 0)) > float(config['SESSION_TIMEOUT']):
            session.clear()
            flash('Сесія закінчилась. Будь ласка, увійдіть знову.', 'warning')
            return redirect(url_for('login'))

        session['last_active'] = now
        return f(*args, **kwargs)
    return decorated_function
```

Рисунок 3.5 – Метод, що встановлює час сесії і перевіряє чи увійшов користувач

Деякі додаткові маршрути було винесено у інший клас, для масштабування та зручності написання коду (Рис 3.6).

```python
class Router():

    def __init__(self, app, user_model, logger):

        self.app = app
        self.user_model = user_model

        host_router = HostRouter()
        scan_router = ScanRouter()

        self.app.register_blueprint(host_router.hosts_router)
        self.app.register_blueprint(scan_router.scan_router)

        self.logger = logger
```

Рисунок 3.6 – Конструктор класу з основними маршрутами

3. Scan routers – клас, що описує правила маршрутизації, які будуть застосовані під час виконання функцій сканування мережевих елементів інформаційної системи. В цьому класі було підключено моделі для роботи з

таблицями результатів сканування у базі даних. Тут прописана логіка обробки деяких POST і GET запитів (Рис. 3.7).

```
class ScanRouter:

    def __init__(self):

        self.scan_router = Blueprint('scan_router', __name__)
        self.setup_routes()

    def setup_routes(self):

        @self.scan_router.route('/scanning')
        def get_all_hosts():

            host_model = HostModel()
            hosts = host_model.get_all_hosts()
            return render_template('scanning.html', hosts=hosts)

        @self.scan_router.route('/scanning/run', methods=['POST'])
        def run_scan():
            host_id = request.form.get('host')
            host_ip = request.form.get(f'ip_{host_id}')
            analizer = NetScanner()
```

Рисунок 3.7 – Фрагмент коду з класу, що описує правила маршрутизації для функцій сканування

4. Host router – клас, що містить правила маршрутизації, які відносяться до функціоналу роботи з хостати та операціями над ними.

Тут було прописано метод пошуку хоста в базі даних за іменем або IP адресою. Було написано методи для обробки даних, які використовуються для дашбордів. Також в цьому класі було реалізовано методи для додавання, редагування чи видаення цільової хостової системи (Рис. 3.8).

```
class HostRouter:

    def __init__(self):
        self.host_model = HostModel()
        self.hosts_router = Blueprint('hosts_router', __name__)
        self.setup_routes()

    def setup_routes(self):

        @self.hosts_router.route('/inventory')

        def get_all_hosts():

            hosts = self.host_model.get_all_hosts()
            return render_template('inventory.html', hosts=hosts)

        @self.hosts_router.route('/inventory/delete/<int:host_id>', methods=['GET','POST'])
        def delete_host(host_id):

            try:
                host_id = int(host_id)
            except ValueError:
                return {"success": False, "errors": ["Некорректный ID хоста"]}


            existing_host = self.host_model.get_host_by_id(host_id)
            if not existing_host:
                return {"success": False, "errors": ["Хост не знайдено"]}
```

Рисунок 3.8 – Фрагмент коду з класу host router

5. Net scan controller – контролер, який реалізовує функціонал аналізу мережевих сокетів. Метод здатен визначати статус сокета, наявність шифрування на сокеті, використаний алгоритм шифрування, терміни видачі та придатності сертифікату, який використовується для шифрування трафіку на вказаному сонеті (Рис. 3.9).

```
for port in range(min_value, max_value+1):
    try:

        with socket.create_connection((host_ip, port), timeout=3) as sock:
            with context.wrap_socket(sock, server_hostname=host_ip) as ssock:

                cert_bin = ssock.getpeercert(binary_form=True)
                cert_obj = x509.load_der_x509_certificate(cert_bin, default_backend())

                tls_info = {
                    "protocol": ssock.version(),
                    "cipher": ssock.cipher(),
                    "issued_to": str(cert_obj.subject),
                    "issued_by": str(cert_obj.issuer),
                    "valid_from": cert_obj.not_valid_before.strftime('%d:%m:%Y %H:%M:%S'),
                    "valid_until": cert_obj.not_valid_after.strftime('%d:%m:%Y %H:%M:%S')
                }


                self.scan_model.save_net_scan_result(
                    scan_id=scan_id,
                    host=host_id,
                    port=port,
                    protocol=str(tls_info["protocol"]),
                    cipher_suite=str(tls_info["cipher"]),
                    issued_to=str(tls_info["issued_to"]),
                    issued_by=str(tls_info["issued_by"]),
                    valid_from=str(tls_info["valid_from"]),
                    valid_until=str(tls_info["valid_until"]),
                    status="secure",
                    error_message="None",
                    created_at=datetime.now().date()
                )
```

Рисунок 3.9 – Фрагмент коду, що відповідає за сканування списку портів

Спочатку, за допомогою бібліотеки SSL було створено unverified context (Рис. 3.10).

```
context = ssl.create_default_context()
context.check_hostname = False
context.verify_mode = ssl.CERT_NONE
```

Рисунок 3.10 – Створення SSL контексту

Цей крок був необхідний для того, щоб python міг пройти SSL handshake і отримати дані про сертифікат кінцевої системи, яка сканується.

Далі було описано цикл, у якому python проходить по списку мережевих портів і до кожного робить підключення на рівні протоколу TCP.

Для цього було використано бібліотеку socket (Рис. 3.11).

```python
for port in range(min_value, max_value+1):
    try:

        with socket.create_connection((host_ip, port), timeout=3) as sock:
            with context.wrap_socket(sock, server_hostname=host_ip) as ssock:

                cert_bin = ssock.getpeercert(binary_form=True)
                cert_obj = x509.load_der_x509_certificate(cert_bin, default_backend())

                tls_info = {
                    "protocol": ssock.version(),
                    "cipher": ssock.cipher(),
                    "issued_to": str(cert_obj.subject),
                    "issued_by": str(cert_obj.issuer),
                    "valid_from": cert_obj.not_valid_before.strftime('%d:%m:%Y %H:%M:%S'),
                    "valid_until": cert_obj.not_valid_after.strftime('%d:%m:%Y %H:%M:%S')
                }
```

Рисунок 3.11 – Фрагмент коду аналізу сокетів зі списку

Для того, щоб дістати інформацію про сертифікати, було використано метод x509 з бібліотеки cryptography.

Цей метод дозволяє представити дані сертифікату у бінарному вигляді, де дозволяє отримати такі дані як алгоритм шифрування (cipher), який було використано для генерації сертифікату (Рис. 3.12).

Код, який створює підключення до сонету –

socket.create_connection((host_ip, port), timeout=3)

Код, який огортає tcp підключення в SSL context –

context.wrap_socket(sock, server_hostname=host_ip)

```python
self.scan_model.save_net_scan_result(
    scan_id=scan_id,
    host=host_id,
    port=port,
    protocol=str(tls_info["protocol"]),
    cipher_suite=str(tls_info["cipher"]),
    issued_to=str(tls_info["issued_to"]),
    issued_by=str(tls_info["issued_by"]),
    valid_from=str(tls_info["valid_from"]),
    valid_until=str(tls_info["valid_until"]),
    status="secure",
    error_message="None",
    created_at=datetime.now().date()
)
```

Рисунок 3.12 – Запис результатів сканування до бази даних.

6. Dos scan controller – клас, що реалізовує функціонал для аналізу кінцевоої системи на відмовостійкість до атак типу DOS, DDOS. Клас, що тут

описаний, здатен надсилати вказану кількість HTTP - запитів на кінцеву хостову систему і аналізувати коди відповіді.

Цей клас отримує бажану кількість запитів користувача і створює таку кількість http запитів до цільової системи (Рис. 3.13, 3.14).

```
try:
    with concurrent.futures.ThreadPoolExecutor(max_workers=self.max_workers) as executor:
        futures = []
        for _ in range(num_requests):
            for p in ports:
                protocol = "https" if use_https or p in [443, 10443, 10444] else "http"
                base_url = f"{protocol}://{host_ip}:{p}"
                path = random.choice(paths)
                url = f"{base_url}{path}"
                futures.append(executor.submit(self._safe_send_request, url))

        for future in concurrent.futures.as_completed(futures):
            result = future.result()
            if result:
                status_code, response_time = result
                self._update_results(results, status_code, response_time)

except (requests.RequestException, concurrent.futures.ThreadPoolExecutor) as e:
    logging.error(f"Помилка при виконанні запитів: {e}")

except Exception as e:
    logging.exception(f"Не відома помилка: {e}")
finally:
    monitor_stop.set()
    monitor_thread.join(timeout=2)
```

Рисунок 3.13 – Фрагмент коду, в якому здійснюються http запити до цільової системи

```
def _evaluate_vulnerability(self, results) -> float:

    score = 5.0


    if results['total_requests'] == 0:
        return score

    if results['failed_requests'] > 0:
        failure_rate = results['failed_requests'] / results['total_requests']
        score += failure_rate * 5

    if results['avg_response_time'] > 1.0:
        score += min(3, results['avg_response_time'] - 1)

    if 503 in results['status_codes']:
        score += 2

    return max(0, min(15, score))
```

Рисунок 3.14 – Фрагмент коду, в якому здійснюється попередня оцінка вразливості

7. Scan model – клас, що описує взаємодію з базою даних. Цей функціонал використовують контролери та роутер, який описує маршрути для функцій сканування.

8. Host model – клас, що описує взаємодію з базою даних. Цей клас використовується для операцій з хостами в базі даних.

9. User model – клас, що використовується для операцій над обліковими записами користувачів та роботою з ними в базі даних.

Усі моделі містять шаблони запитів до бази, які використовуються веб додатком (Рис 3.15).

```
def search_hosts(self, search_term):

    query = """
    SELECT id, hostname, ip_address, os_type, status
    FROM hosts
    WHERE hostname LIKE %s OR ip_address LIKE %s
    ORDER BY hostname
    """
    search_pattern = f"%{search_term}%"
    return self.db.fetch_all(query, (search_pattern, search_pattern))
```

Рисунок 3.15 – Фрагмент коду з класу моделі, для пошуку системи по ID

10. Main db connector – клас, що описує підключення до бази даних і семантичні шаблони взаємодії з таблицями в реляційній базі (Рис 3.16).

```
class Database:

    def __init__(self):

        self.connection = None
        self.cursor = None
        self.connect()

    def connect(self):

        with open("cfg/config.yml", 'r') as f:
            config = yaml.safe_load(f)
            db_config = config['DATABASE_CONFIG']
        try:
            self.connection = mysql.connector.connect(
                host=db_config['host'],
                database=db_config['database'],
                user=db_config['user'],
                password=db_config['password']
            )

            if self.connection.is_connected():
                logging.info("Successfully connected to MySQL database")
                self.cursor = self.connection.cursor(dictionary=True)
        except Error as e:
            logging.error(f"Error connecting to MySQL database: {e}")

    def disconnect(self):

        if self.connection and self.connection.is_connected():
            self.cursor.close()
            self.connection.close()
            logging.info("MySQL connection closed")
```

Рисунок 3.16 – Метод що описує підключення до бази даних

Подібна архітектура веб додаку є масштабованою, простою в обслуговуванні а також надамає можливість тестування окремих класів додатку окремо від користувача. Це сприяє підвищенню відмовостійкості роботи додаку у продуктивному середовищі.

Інтерфейс користувача

За замовчуванням додаток запускається на адресі localhost : 5000

Спершу відкриється сторінка логіну (Рис. 3.17).

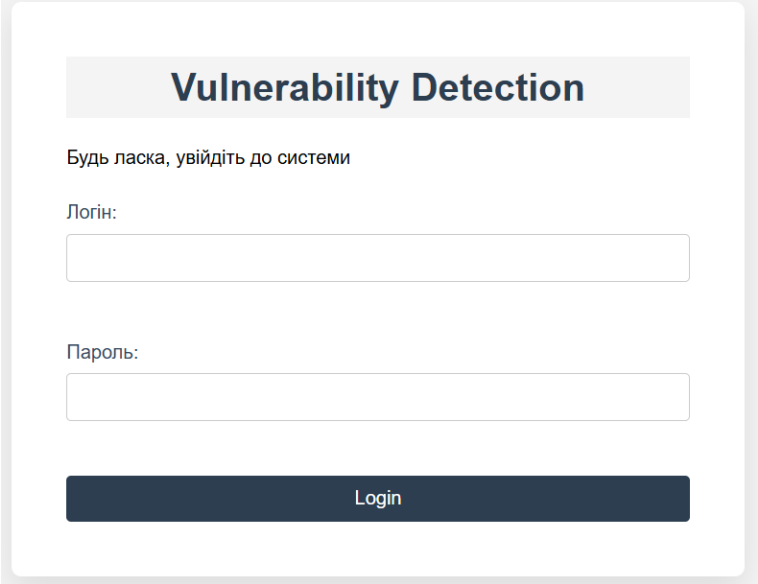

Рисунок 3.17 – Вікно входу до веб додатку

Для зручності роботи з хостами, їх додаванням, редагуванням, було написано окрему вкладку меню. Для структурування, інформацію було поміщено у таблицю (Рис. 3.18, 3.19).

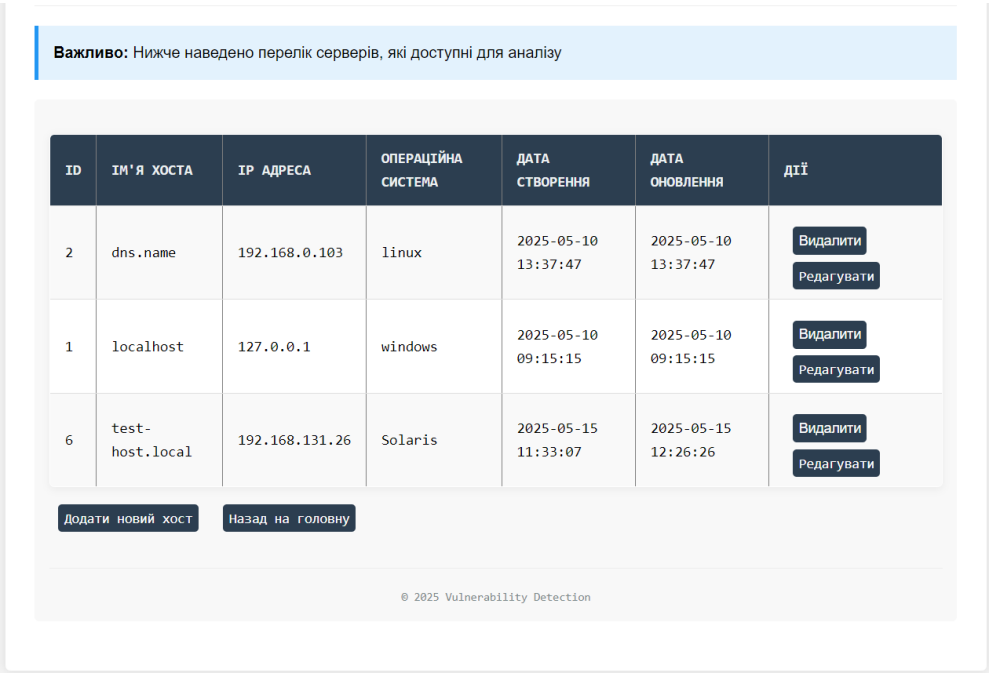

Рисунок 3.18 – Вікно додавання цільових систем

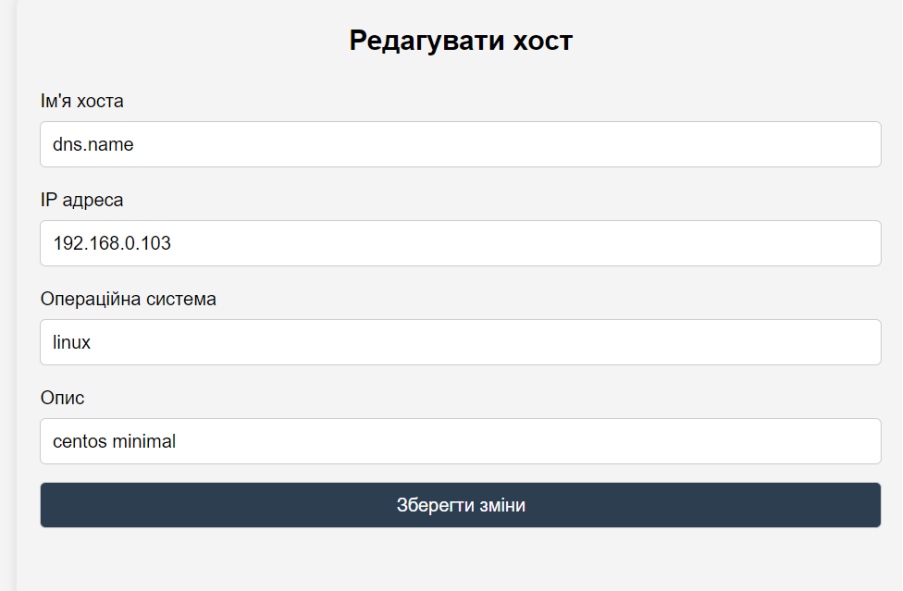

Рисунок 3.19 – Вікно редагування інформації про цільову систему

На сторінку «Сканування» було додано 2 вкладки з доступними наразі типами сканування.

Для сканування портів – можна вказати діапазон бажаних портів (Рис. 3.20).

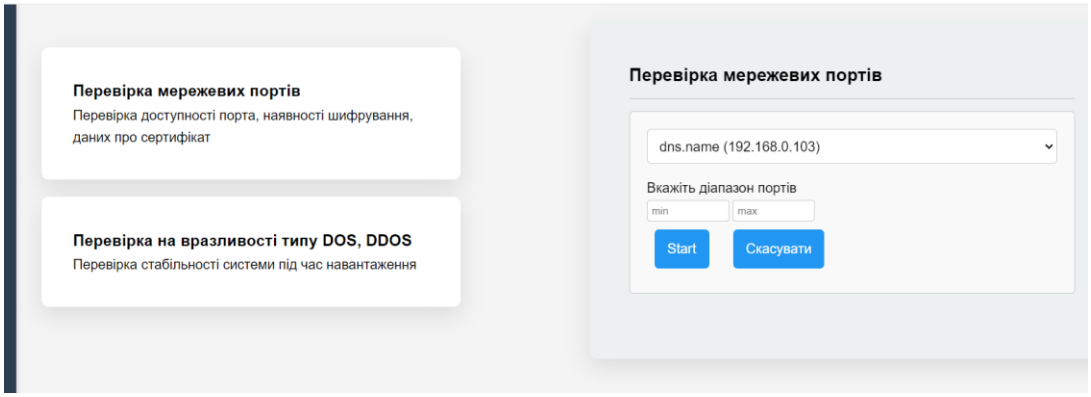

Рисунок 3.20 – Сторінка сканування, вкладка «Перевірка мережевих портів»

Для вкладки «Перевірка на вразливості типу DOS, DDOS» було додано можливість обирати кількість запитів, які буде відправлено на цільову систему (Рис. 3.21).

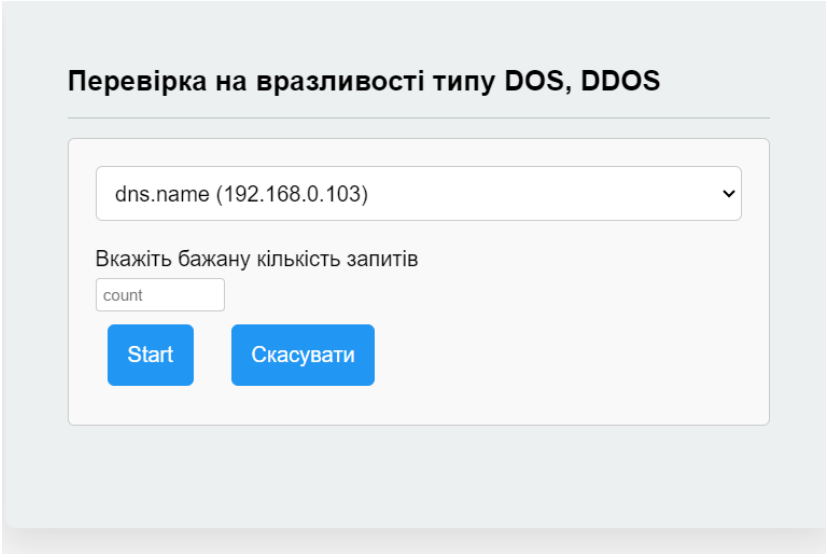

Рисунок 3.21 – Вкладка сканування на вразливості типу DOS, DDOS

На сторінці сканування було додано можливість збереження результатів сканування у PDF формат.

Для відображення даних сканування типу DDOS було розроблено сторінку з деякими графіками. Цей функціонал передбачає запит до бази даних і вибірку інформації по ID хоста (Рис. 3.22, 3.23, 3.24).

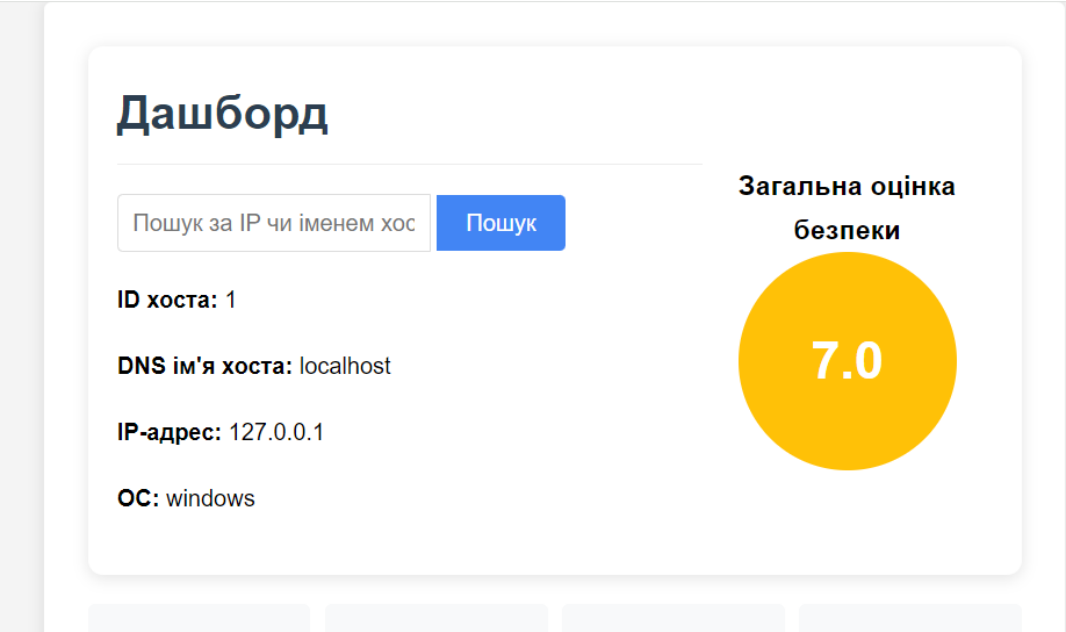

Рисунок 3.22 – Сторінка відображення даних про хост і попередня оцінка

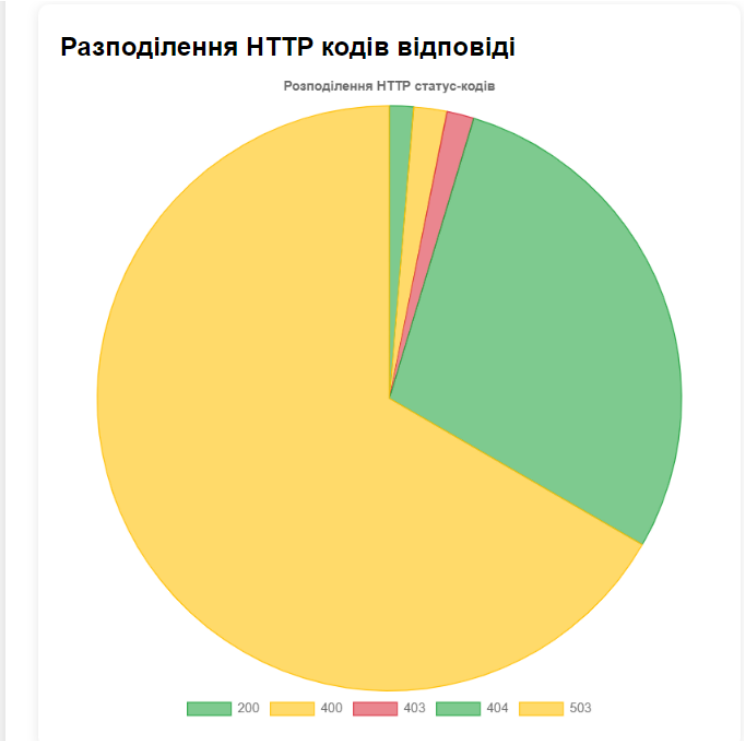

Рисунок 3.23 – Графік розподілення HTTP кодів відповіді

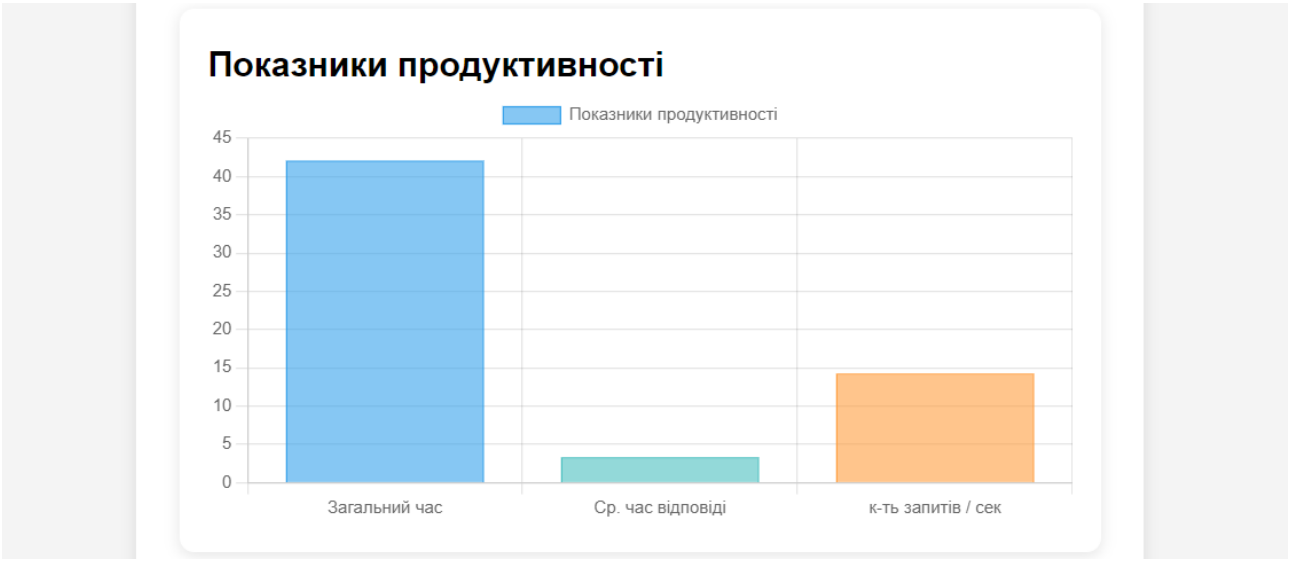

Рисунок 3.24 – Показники продуктивності

Для зручності змінювання деяких налаштувань було додано окрему сторінку, на яку виводяться параметри з конфігураційного файлу. Конфігураційний файл зберігається у форматі YAML (Рис. 3.25).

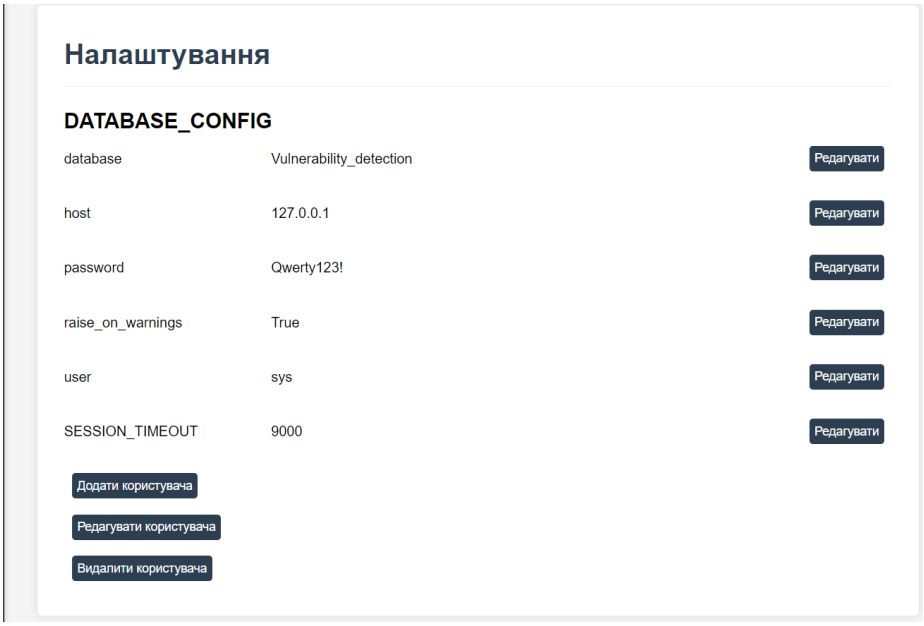

Рисунок 3.25 – Сторінка відображення налаштувань

Для зручності орієнтування в інтерфейсі веб додатку було додано вертикальне меню з лівого боку. Цей функціонал додає зручності у навігації в графічному інтерфейсі (Рис. 3.26).

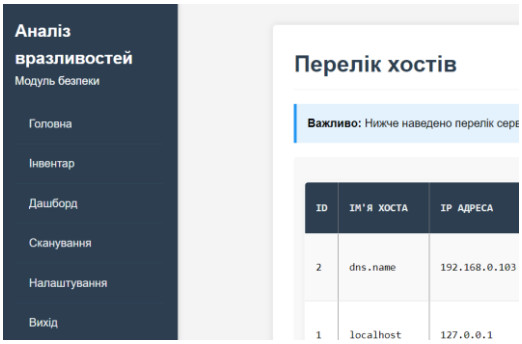

Рисунок 3.26 – Вертикальне меню з лівого боку

Для роботи з користувачами було додано 3 кнопки у вкладці «Налаштування».

Функціонал роботи з користувачами наразі дозволяє додавати, видаляти або змінювати таку інформацію про користувача як пароль, логін або адреса електронної пошти (Рис. 3.27, 3.28).

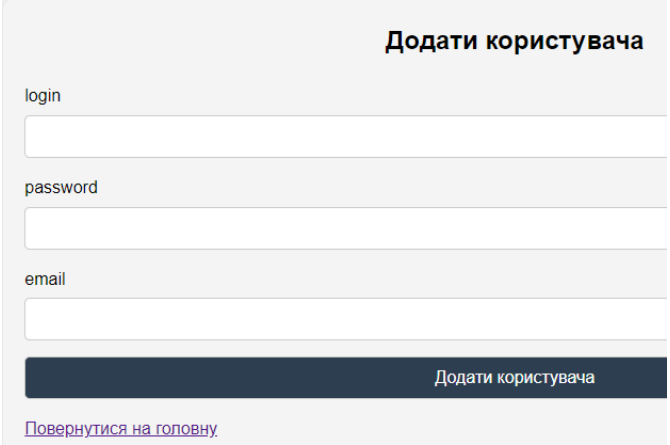

Рисунок 3.27 – Вікно додавання користувача

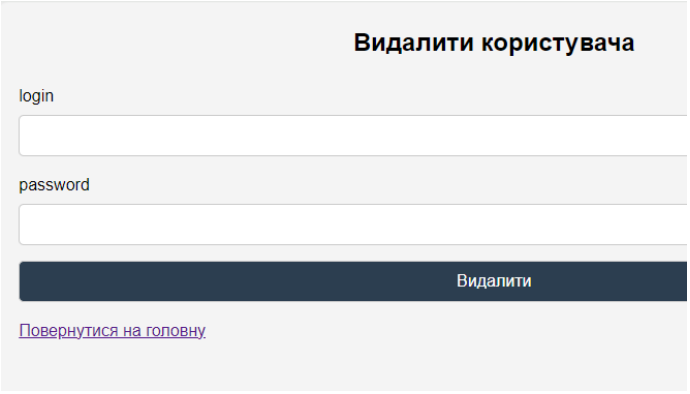

Рисунок 3.28 – Вікно видаляння користувача

3.4 Інструкція користувача

Щоб налаштувати оточення для роботи додатку необхідно встановити python, flask та бібліотеки з розділу 3.1.

За замовчуванням доадаток запускається на адресі 127.0.0.1 5000

Адресу необхідно відкрити в браузері.

Облікові дані за замовчуванням :

Лоін для входу – admin

Пароль – password

Ці дані створюються автоматично при першому запуску. Надалі можна змінити пароль або логін.

Орієнтуватись у додатку можна за допомогою вертикального меню з лівого боку.

Спершу необхідно додати цільову систему, яку необхідно сканувати. Це можна зробити у вкладці інвентар. Обов’язковими полями є IP адреса та ім’я системи.

Для початку сканування – необхідно перейти на вкладку «Сканування» та обрати тип сканування з доступного переліку методів.

Після завершення сканування буде відображено таблицю з відповідними результатами. Поряд є кнопка, яка дозволяє зберегти таблицю с результатами сканування у PDF файл.

У вкладці дашборд можна скористатися пошуковою строкою і знайти цільову систему, за результатами сканування якої можна побачити приблизні графіки.

У якості пропозиції щодо автоматизації встановлення – налаштування docker контейнеру з необхідними залежностіми.

Для зміни налаштувань веб додатку або роботи з користувачами можна використовувати вкладку – «Налаштування».

3.5 Тестування програмного продукту

При додаванні нового хоста – не можна лишати пусті поля (Рис. 3.29).

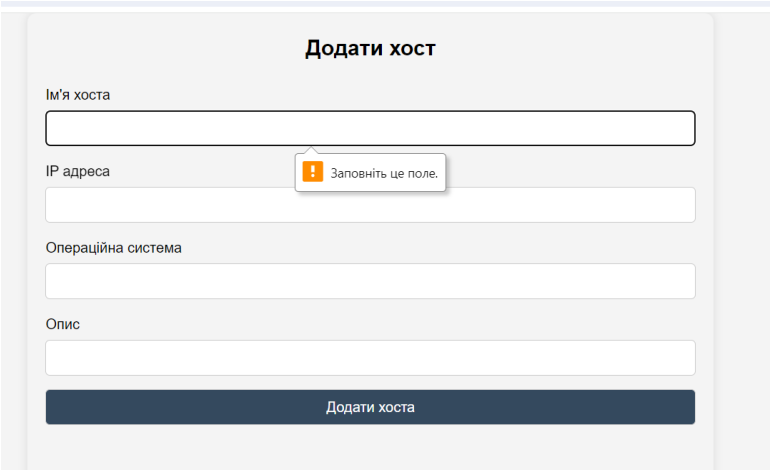

Рисунок 3.29 – Попередження про не можливість додавання цільової системи з пустим полем імені

Перш ніж вийти з додатку, система спитає чи дійсно користувач цього бажає (Рис. 3.30).

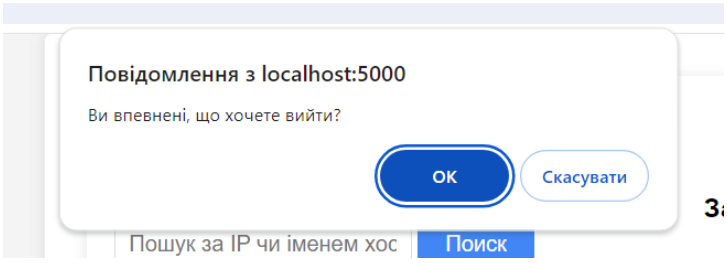

Рисунок 3.30 – Попередження про вихід

Під час входу до додатку не можна лишати пусті поля логіну чи паролю, або вводити не правильний пароль.

На сторінці можна побачити відповідні попередження (Рис. 3.31, 3.32).

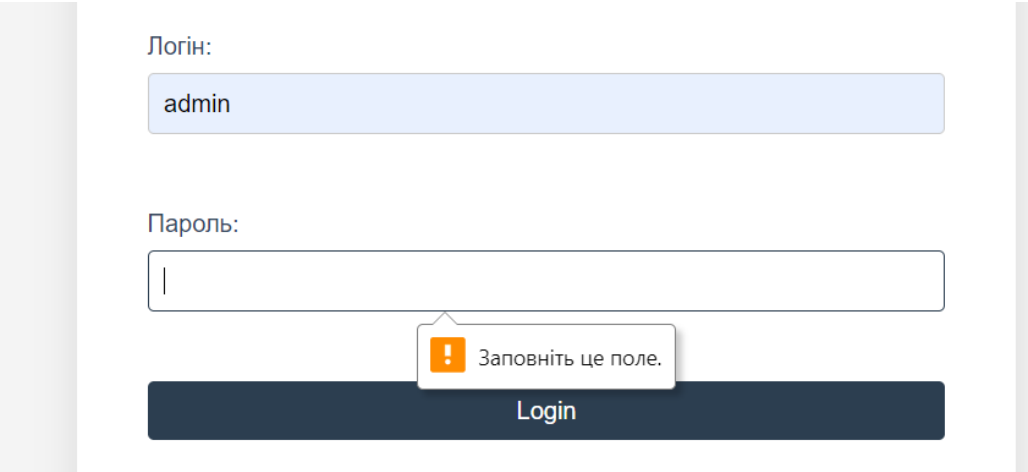

Рисунок 3.31 – Попередження про пусте поле паролю

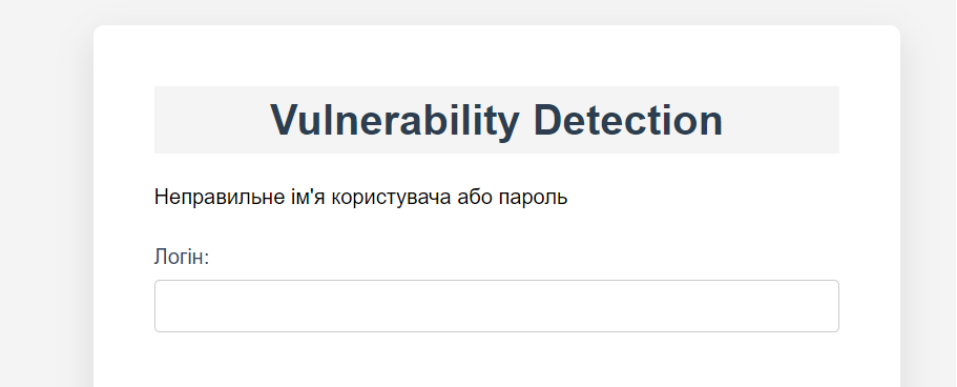

Рисунок 3.32 – Попередження про не правильний логін або пароль

Розміри екрану можуть бути зменшені до 792 px

Використання додатку передбачається тільки через корпоративний VPN , тому використання його на пристроях, екран яких має менше розширення ніж планшетний – не передбачено (Рис. 3.33).

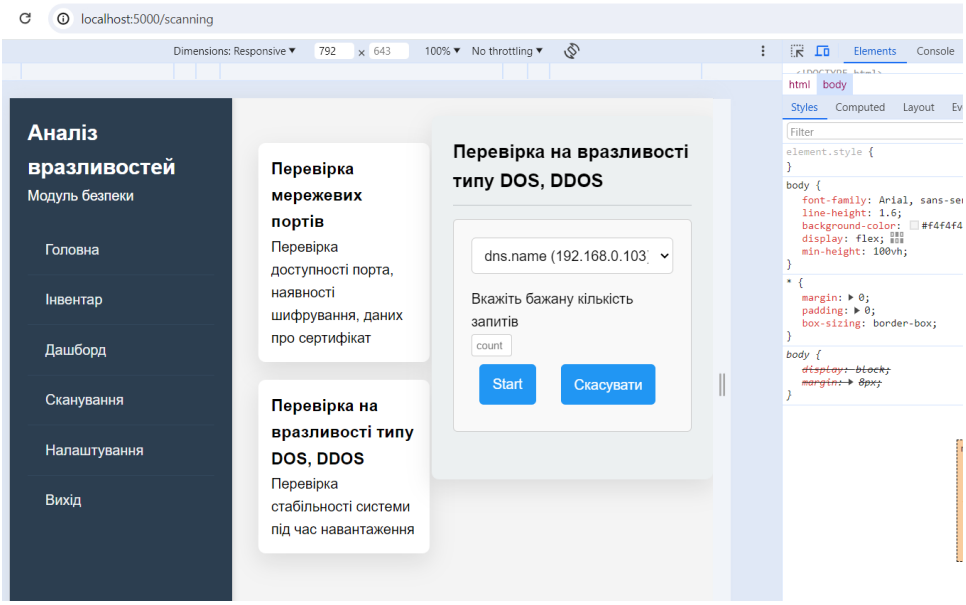

Рисунок 3.33 – Інтерфейс додатку з ввімкненим вікном розробника для тестування зменшення розміру екрану

Веб додаток цілком є кросплатформеним і може бути запущений у різних браузерах. Було протестовано відображення графічного інтерфейсу у Mozilla firefox (Рис. 3.34, 3.35) .

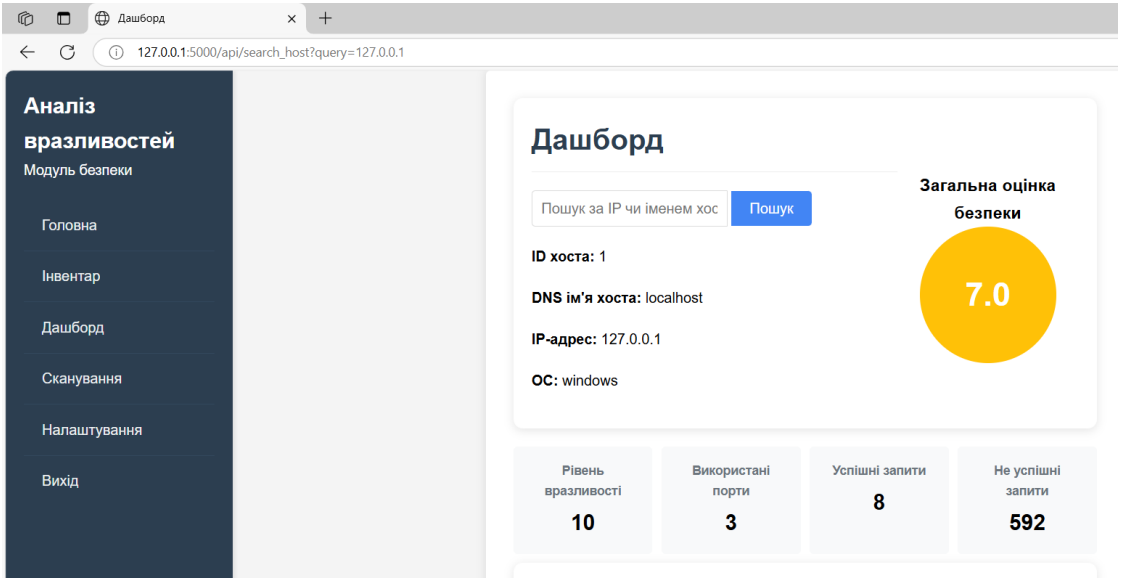

Рисунок 3.34 Вкладка дашборд у браузері mozilla firefox

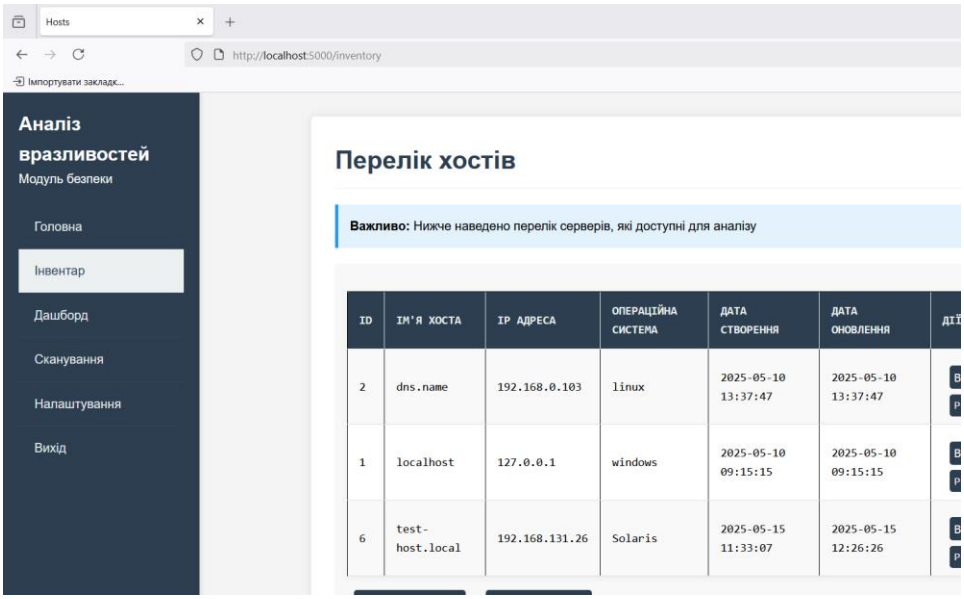

Рисунок 3.35 Сторінка з відображенням цільових систем у браузері mozilla firefox

Було протестовано функціональність графічного інтерфейсу у браузері Microsoft edge (Рис. 3.36).

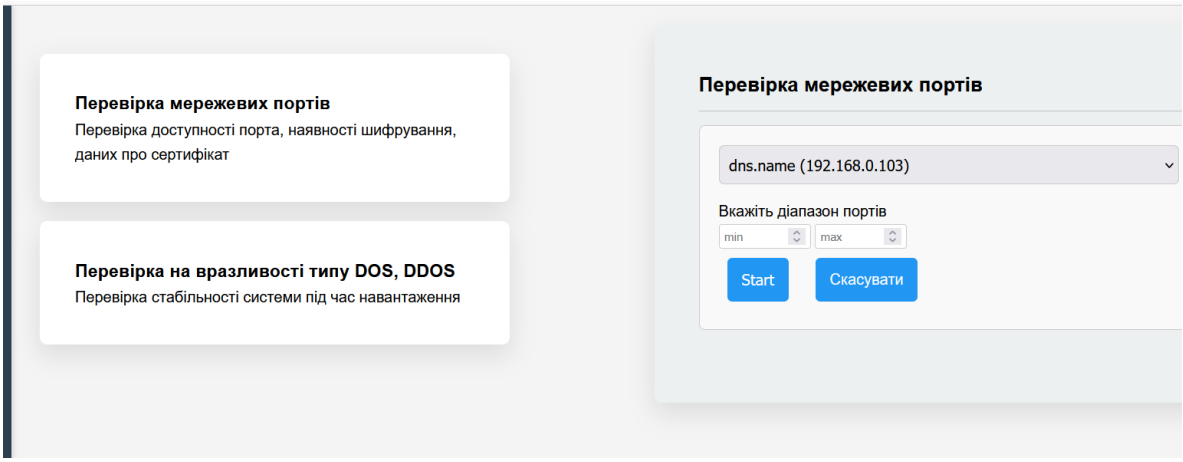

Рисунок 3.36 Вкладка з відображенням сканування мережевих портів у браузері Microsoft edge

ВИСНОВКИ

У даній дипломній роботі було розроблено та реалізовано модуль аналізу інформаційних систем на наявність вразливостей. Було розроблено програмний інструмент, що дозволяє виявити потенційні загрози безпеці інформаційних систем та мереж.

Розробка модуля здійснювалася з використанням мови програмування Python та фреймворку Flask, що забезпечило гнучкість розробки та можливість швидкого розгортання застосунку в різних середовищах. Вибір цих технологій був обумовлений їхньою потужністю, великою кількістю доступних бібліотек та широкою спільнотою розробників.

У процесі роботи над дипломним проєктом були виконані наступні завдання:

- Проаналізовано сучасні методи та підходи до виявлення вразливостей інформаційних систем. На основі аналізу літературних джерел та існуючих рішень було виявлено основні напрямки, за якими доцільно проводити перевірку захищеності систем.
- Досліджено особливості реалізації методів аналізу відкритих портів та стійкості до DDoS-атак.
- Визначено ключові параметри, що підлягають перевірці, та розроблено алгоритми їх аналізу.
- Розроблено архітектуру модуля, що забезпечує гнучкість, масштабованість та можливість подальшого розширення функціоналу.
- Реалізовано метод аналізу відкритих портів, який включає детальне дослідження SSL/TLS сертифікатів, виявлення версій протоколів, аналіз використаних алгоритмів шифрування, перевірку емітентів сертифікатів та термінів їх дії.
- Створено зручний користувацький інтерфейс, що надає можливість додавання, редагування та видалення хостів для аналізу, управління користувачами системи та налаштування параметрів сканування.

Розроблений модуль може знайти практичне застосування в процесах аудиту безпеки інформаційних систем, в підрозділах інформаційної безпеки організацій різного масштабу, а також у навчальному процесі при підготовці фахівців з кібербезпеки.

Важливо відзначити, що запропоноване рішення не є статичним і передбачає можливість подальшого розвитку та вдосконалення.

Таким чином, результати виконаної дипломної роботи мають як теоретичну, так і практичну цінність. Теоретична значущість полягає в систематизації підходів до аналізу вразливостей інформаційних систем та розробці методології їх виявлення. Практична цінність роботи визначається створенням функціонального програмного продукту, який може бути використаний для реального підвищення рівня безпеки інформаційних систем.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ ІНФОРМАЦІЇ

1.Методичні рекомендації до виконання кваліфікаційної роботи . – [Електронний ресурс]. – Режим доступу: https://cde.nuft.edu.ua/mod/resource/view.php?id=751619

2.Офіційна документація мови python . – [Електронний ресурс]. – Режим доступу: https://docs.python.org/3/.

3.Офіційна документація фреймворку flask . – [Електронний ресурс]. – Режим доступу: https://flask.palletsprojects.com/en/stable/

4.Офіційна документація бібліотеки python SSL . – [Електронний ресурс]. – Режим доступу: https://docs.python.org/3/library/ssl.html

5.Офіційна документація бібліотеки python socket . – [Електронний ресурс]. – Режим доступу: https://docs.python.org/3/library/socket.html

6.Документація від Microsoft про визанчення атак типу DDOS . – [Електронний ресурс]. – Режим доступу: https://www.microsoft.com/uk-ua/security/business/security-101/what-is-a-ddos attack

7.Документація по мові Javascript . – [Електронний ресурс]. – Режим доступу: https://uk.javascript.info/

8.Документація MySql –. – [Електронний ресурс]. – Режим доступу: https://dev.mysql.com/doc/

9. Документація для розуміння транспортного рівня мережевої моделі OSI –. – [Електронний ресурс]. – Режим доступу: https://www.geeksforgeeks.org/transport-layer-in-osi-model/

10. Найкращі практики безпеки в MySQL . – [Електронний ресурс]. – Режим доступу: Best Practices For MySQL Security | GeeksforGeeks

11. Рекомендації безпеки для фланк додатків . – [Електронний ресурс]. – Режим доступу: https://www.securecoding.com/blog/flask-security-best-practices/

12. Зростання DDoS-атак у 2023 році . – [Електронний ресурс]. – Режим доступу: https://www.axios.com/2023/07/07/ddos-cyberattack-more-destructive
13. Рекомендації OWASP щодо безпеки TLS . – [Електронний ресурс]. – Режим доступу: https://cheatsheetseries.owasp.org/cheatsheets/Transport_Layer_Security_Cheat_Sheet.html
14. Небезпечні шифри та їх вразливості . – [Електронний ресурс]. – Режим доступу: https://www.wired.com/story/uk-banks-transport-layer-security
15. Бібліотека python cryptography . – [Електронний ресурс]. – Режим доступу: https://cryptography.io/en/latest/index.html
16. Приклад використання симетричного шифрування з python cryptography . – [Електронний ресурс]. – Режим доступу: https://cryptography.io/en/latest/index.html
17. Опис протоколу HTTP . – [Електронний ресурс]. – Режим доступу: https://developer.mozilla.org/en-US/docs/Web/HTTP/Reference/Status
18. Документація протоколу SSL \ TLS . – [Електронний ресурс]. – Режим доступу: Transport Layer Security (TLS) - Security on the web | MDN
19. Повний посібник з розробки фінтех-програмного забезпечення 2024. URL: https://youteam.io/blog/complete-guide-to-fintech-software-development/ (дата звернення: 12.05.2025)
20. DashDevs. Роботизована автоматизація процесів у фінтех та банківському секторі. URL: https://dashdevs.com/blog/robotic-process-automation-is-to-become-a-growth-strategy-for-the-financial-industry/ (дата звернення: 15.05.2025)
21. Workato. Правильний підхід до автоматизації процесів для фінтех. URL: https://www.workato.com/the-connector/process-automation-for-fintech/ (дата звернення: 18.05.2025)

22. Kleppmann, M. Designing Data-Intensive Applications: The Big Ideas Behind Reliable, Scalable, and Maintainable Systems. O'Reilly Media, 2017. 616 p. (дата звернення: 19.05.2025)
23. Chishti, S., Barberis, J. The FINTECH Book: The Financial Technology Handbook for Investors, Entrepreneurs and Visionaries. Wiley, 2016. 304 p. (дата звернення: 17.05.2025)
24. Національний банк України. Розділ "Платіжні системи та інноваційний розвиток". URL: https://bank.gov.ua/ua/promo/payments-innovations (дата звернення: 14.05.2025)
25. ISO 20022. Official ISO 20022 Website. URL: https://www.iso20022.org (дата звернення: 20.05.2025)

ДОДАТКИ

Додаток А – Код методу сканування портів

Лістинг коду з файлу net_scsanner.py (логіка сканування портів)

```
import socket
import time
import logging
import ssl
from models.scan_model import ScanModel
from datetime import datetime
from cryptography import x509
from cryptography.hazmat.backends import default_backend

class NetScanner:
    def __init__(self):
        self.scan_model = ScanModel()

    def net_scan_run(self, host_id, host_ip, min_value, max_value):

        scan_id = self.scan_model.create_scan(host_id=host_id, scan_type="port_scan", created_by="admin")

        tls_results = []


        context = ssl.create_default_context()
        context.check_hostname = False
        context.verify_mode = ssl.CERT_NONE


        for port in range(min_value, max_value+1):
            try:

                with socket.create_connection((host_ip, port), timeout=3) as sock:
                    with context.wrap_socket(sock, server_hostname=host_ip) as ssock:
```

```
                cert_bin = ssock.getpeercert(binary_form=True)
                cert_obj = x509.load_der_x509_certificate(cert_bin, default_backend())

                tls_info = {
                    "protocol": ssock.version(),
                    "cipher": ssock.cipher(),
                    "issued_to": str(cert_obj.subject),
                    "issued_by": str(cert_obj.issuer),
                    "valid_from": cert_obj.not_valid_before.strftime('%d:%m:%Y %H:%M:%S'),
                    "valid_until": cert_obj.not_valid_after.strftime('%d:%m:%Y %H:%M:%S')
                }


                self.scan_model.save_net_scan_result(
                    scan_id=scan_id,
                    host=host_id,
                    port=port,
                    protocol=str(tls_info["protocol"]),
                    cipher_suite=str(tls_info["cipher"]),
                    issued_to=str(tls_info["issued_to"]),
                    issued_by=str(tls_info["issued_by"]),
                    valid_from=str(tls_info["valid_from"]),
                    valid_until=str(tls_info["valid_until"]),
                    status="secure",
                    error_message="None",
                    created_at=datetime.now().date()
                )

                tls_results.append((port, "secured", tls_info))

                logging.info(f"TLS scan completed for {host_ip}:{port}")

            except socket.timeout:
                status = "timeout"
                error_message = "Connection timeout"
                logging.warning(f"TLS scan timed out on port {port} for {host_ip}")

            except ssl.SSLError as e:
                status = "ssl_error"
                error_message = str(e)
```

```
            tls_results.append((port, "ssl_error", str(e)))
            self.scan_model.save_net_scan_result(
                    scan_id=scan_id,
                    host=host_id,
                    port=port,
                    protocol=str("ssl error"),
                    cipher_suite=str("No ciphers"),
                    issued_to=str("No data"),
                    issued_by=str("No data"),
                    valid_from=None,
                    valid_until=None,
                    status="No secure",
                    error_message=str(e),
                    created_at=datetime.now().date()
                )
            logging.warning(f"SSL error on port {port} for {host_ip}: {e}")

        except ConnectionRefusedError:
            status = "closed"
            logging.info(f"Connection refused on port {port} for {host_ip}")

        except Exception as e:
            status = "error"
            error_message = str(e)
            tls_results.append((port, "error", str(e)))
            logging.error(f"TLS scan failed on port {port} for {host_ip}: {e}")

        time.sleep(0.5)

    return tls_results

def run_scan(self):

    print("SQL Injection Scanner for VirtualBox Machines")
    print("=" * 50)

    # Scan ports
    open_ports = self.scan_ports()

    if not open_ports:
```

```
        print("\n[!] No open ports found to test")
        return

    print(f"\n[+] Found {len(open_ports)} open ports: {open_ports}")

    vulnerable_ports = []

    for port in open_ports:
        if self.test_sql_injection(port):
            vulnerable_ports.append(port)

    print("\n" + "=" * 50)
    print("Scan Summary:")
    print(f"Total open ports: {len(open_ports)}")
    print(f"Potentially vulnerable ports: {len(vulnerable_ports)}")

    if vulnerable_ports:
        print(f"Vulnerable ports: {vulnerable_ports}")
        print("\n[!] Warning: These results are based on basic testing and may include false positives.")
        print("[!] Manual verification is recommended for accurate assessment.")
    else:
        print("\n[+] No obvious SQL injection vulnerabilities were detected.")
        print("[+] Note: This does not guarantee the absence of more sophisticated vulnerabilities.")
```

Додаток Б – Код методу моделі підключення до бази данних

Лістинг коду з файлу _db.py (логіка підключення до бази та опис семантики запитів)

```
import mysql.connector
from mysql.connector import Error
import logging
import yaml

class Database:

    def __init__(self):

        self.connection = None
        self.cursor = None
        self.connect()

    def connect(self):

        with open("cfg/config.yml", 'r') as f:
            config = yaml.safe_load(f)
            db_config = config['DATABASE_CONFIG']
        try:
            self.connection = mysql.connector.connect(
                host=db_config['host'],
                database=db_config['database'],
                user=db_config['user'],
                password=db_config['password']
            )

            if self.connection.is_connected():
                logging.info("Successfully connected to MySQL database")
                self.cursor = self.connection.cursor(dictionary=True)
        except Error as e:
            logging.error(f"Error connecting to MySQL database: {e}")

    def disconnect(self):
```

```
    if self.connection and self.connection.is_connected():
        self.cursor.close()
        self.connection.close()
        logging.info("MySQL connection closed")

def execute_query(self, query, params=None):

    try:
        if not self.connection or not self.connection.is_connected():
            self.connect()

        self.cursor.execute(query, params or ())
        self.connection.commit()
        return True
    except Error as e:
        logging.error(f"Error executing query: {e}")
        return False

def fetch_all(self, query, params=None):

    try:
        if not self.connection or not self.connection.is_connected():
            self.connect()

        self.cursor.execute(query, params or ())
        return self.cursor.fetchall()
    except Error as e:
        logging.error(f"Error fetching data: {e}")
        return []

def fetch_one(self, query, params=None):

    try:
        if not self.connection or not self.connection.is_connected():
            self.connect()

        self.cursor.execute(query, params or ())
        return self.cursor.fetchone()
    except Error as e:
        logging.error(f"Error fetching data: {e}")
```

```
        return None

def insert(self, query, params=None):

    try:
        if not self.connection or not self.connection.is_connected():
            self.connect()

        self.cursor.execute(query, params or ())
        self.connection.commit()
        return self.cursor.lastrowid
    except Error as e:
        logging.error(f"Error inserting data: {e}")
        return None

def fetch_query(self, query, params=None):

    try:
        if not self.connection or not self.connection.is_connected():
            self.connect()

        self.cursor.execute(query, params or ())
        result = self.cursor.fetchall()
        return result
    except Error as e:
        logging.error(f"Error fetching data: {e}")
        return None
```

Додаток В – Javascript код, який відображає графіки

Лістинг коду з файлу dashboard.js (логіка відображення графіків)

```
const dashboardDataElement = document.querySelector('script[data-dashboard]');
    if (dashboardDataElement) {
        const dashboardData = JSON.parse(dashboardDataElement.textContent);
        initDashboard(dashboardData);
    }

    function initDashboard(data) {

        document.getElementById("hostId").textContent = data.host_id;
        document.getElementById("hostname").textContent = data.hostname;
        document.getElementById("ipAddress").textContent = data.ip_address;
        document.getElementById("osType").textContent = data.os_type;


        document.getElementById("vulnerabilityScore").textContent = data.vulnerability.vulnerability_score;
        document.getElementById("portsCount").textContent = data.ports.length;
        document.getElementById("successfulRequests").textContent = data.vulnerability.successful_requests;
        document.getElementById("failedRequests").textContent = data.vulnerability.failed_requests;


        const securityScore = calculateSecurityScore(data);
        const scoreElement = document.getElementById("securityScoreCircle");
        scoreElement.textContent = securityScore;
        scoreElement.style.backgroundColor = getScoreColor(securityScore);


        const portsCtx = document.getElementById('portsChart').getContext('2d');
        const protocols = data.ports.map(p => `${p.protocol} (${p.port})`);
        new Chart(portsCtx, {
            type: 'bar',
            data: {
                labels: protocols,
                datasets: [{
                    label: 'Статус порта',
                    data: data.ports.map(p => p.status === "valid" ? 10 : 3),
```

```
                backgroundColor: data.ports.map(p => p.status === "valid" ? 'rgba(40, 167, 69, 0.6)' : 'rgba(220, 53, 69, 0.6)'),
                borderColor: data.ports.map(p => p.status === "valid" ? 'rgb(40, 167, 69)' : 'rgb(220, 53, 69)'),
                borderWidth: 1
            }]
        },
        options: {
            scales: {
                y: {
                    beginAtZero: true,
                    max: 10,
                    title: {
                        display: true,
                        text: 'Попередня оцінка'
                    }
                }
            }
        }
    });


    const vulnCtx = document.getElementById('vulnerabilityChart').getContext('2d');

    const statusCodesStr = data.vulnerability.status_codes;
    const statusCodesMap = {};
    statusCodesStr.split(',').forEach(item => {
        const [code, count] = item.split(':');
        statusCodesMap[code] = parseInt(count);
    });

    new Chart(vulnCtx, {
        type: 'pie',
        data: {
            labels: Object.keys(statusCodesMap),
            datasets: [{
                data: Object.values(statusCodesMap),
                backgroundColor: [
                    'rgba(40, 167, 69, 0.6)',
                    'rgba(255, 193, 7, 0.6)',
                    'rgba(220, 53, 69, 0.6)'
```

```
            ],
            borderColor: [
                'rgb(40, 167, 69)',
                'rgb(255, 193, 7)',
                'rgb(220, 53, 69)'
            ],
            borderWidth: 1
        }]
    },
    options: {
        plugins: {
            title: {
                display: true,
                text: 'Розподілення HTTP статус-кодів'
            },
            legend: {
                position: 'bottom'
            }
        }
    }
});



const perfCtx = document.getElementById('performanceChart').getContext('2d');
new Chart(perfCtx, {
    type: 'bar',
    data: {
        labels: ['Загальний час', 'Ср. час відповіді', 'к-ть запитів / сек'],
        datasets: [{
            label: 'Показники продуктивності',
            data: [
                data.vulnerability.total_time,
                data.vulnerability.avg_response_time,
                data.vulnerability.requests_per_second
            ],
            backgroundColor: [
                'rgba(54, 162, 235, 0.6)',
                'rgba(75, 192, 192, 0.6)',
                'rgba(255, 159, 64, 0.6)'
```

```
                ],
                borderColor: [
                    'rgb(54, 162, 235)',
                    'rgb(75, 192, 192)',
                    'rgb(255, 159, 64)'
                ],
                borderWidth: 1
            }]
        },
        options: {
            scales: {
                y: {
                    beginAtZero: true
                }
            }
        }
    });
}

function calculateSecurityScore(data) {

    let score = 10 - (data.vulnerability.vulnerability_score / 10);

    const failedPorts = data.ports.filter(p => p.status !== "valid").length;
    score -= failedPorts * 0.5;

    const expiredCerts = data.certificates ? data.certificates.filter(c => c.status !== "valid").length : 0;
    score -= expiredCerts * 1;


    const failureRate = data.vulnerability.failed_requests /
        (data.vulnerability.successful_requests + data.vulnerability.failed_requests);
    score -= failureRate * 2;


    return Math.max(0, Math.min(10, score)).toFixed(1);
}


function getScoreColor(score) {
```

```
    if (score >= 8) return "#28a745";
    if (score >= 6) return "#ffc107";
    return "#dc3545";
}



document.querySelector('form').addEventListener('submit', function(event) {
    event.preventDefault();
    const searchQuery = document.getElementById('hostSearch').value.trim();
    if (searchQuery) {
        window.location.href = `/host?query=${encodeURIComponent(searchQuery)}`;
    }
})
```