Хмельницький національний університет
Факультет інформаційних технологій
Кафедра кібербезпеки

КВАЛІФІКАЦІЙНА РОБОТА

Кобець Яни Ростиславівни

на здобуття ступеня вищої освіти Бакалавра

Система виявлення вразливостей вебзастосунків

Галузь знань 12 – Інформаційні технології
Спеціальність 125 – Кібербезпека
Освітня програма Кібербезпека

Шифр КРБКБ.200108.20.01.12 ПЗ

Виконала студентка 4 курсу група КБ-20-1 Яна КОБЕЦЬ
Керівник канд. техн. наук, доцент Вікторія ОРЛЕНКО
Нормоконтролер старший викладач Сергій МОСТОВИЙ

До захисту допускаю:
Завідувач кафедри кібербезпеки Юрій КЛЬОЦ

19 06 2024 р.

Хмельницький 2024

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет Інформаційних технологій

Кафедра Кібезбезпеки

Рівень вищої освіти Перший (бакалаврський)

Галузь знань 12 «Інформаційні технології»

Спеціальність 125 «Кібербезпека»

Освітня програма Кібербезпека

ЗАТВЕРДЖУЮ
Завідувач кафедри кіберзбезпеки
Юрій КЛЬОЦ
15 лютого 2024 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Кобець Яні Ростиславівні

Прізвище, ім'я, по батькові студента

1. Тема роботи Система виявлення вразливостей вебзастосунків

Керівник роботи Вікторія Сергіївна Орленко, канд. техн. наук, доцент

Прізвище, ім'я, по батькові, науковий ступінь, вчене звання

Затверджена наказом ректора університету від 15.02.2024 р. № 8

2. Строк подання студентом проєкту (роботи) на кафедру 13.06.2024

3. Вихідні дані до роботи Створити дієву систему для виявлення вразливості вебзастосунків задля запобігання атак та витоку інформації

4. Зміст пояснювальної записки (перелік питань, які потрібно розробити) Дослідження можливих атак на сайти, опис цих атак та схема роботи задля запобігання їх в подальшому. Дослідження актуальних систем захисту вебзастосунків, розробка та моделювання власної системи виявлення вразливості вебзастосунків, розробка бази даних системи

5. Перелік графічного матеріалу (із зазначенням обов'язкових креслень) «Логічна схема бази даних», «Модель архітектури роботи системи», «Модель вимог до системи виявлення вразливості вебзастосунків»

6. Консультанти розділів кваліфікаційної роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Нормконтроль	Мостовий С.В., Старши викладач кафедри кібербезпеки		

7. Дата видачі завдання « 16 » лютого 2024 р.

КАЛЕНДАРНИЙ ПЛАН

Назва етапів (розділів) кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
Вибір і затвердження теми кваліфікаційної роботи	01.01-01.02.2024	
Ознайомлення з предметною областю	01.02-28.02.2024	
Дослідження існуючих рішень	01.02-28.02.2024	
Постановка задачі	01.03-31.03.2024	
Визначення загальних принципів рішення задачі	01.03-31.03.2024	
Деталізація принципів рішення задачі	01.04-30.04.2024	
Розробка проєктних рішень	01.04-30.04.2024	
Апробація проєктних рішень	01.05-31.05.2024	
Оформлення пояснювальної записки згідно вимог	01.05-31.05.2024	
Оформлення графічної частини	01.06.2024	
Захист КР	01.06.2024	

Студентка _____ Яна КОБЕЦЬ
Підпис Ініціали, прізвище

Керівник кваліфікаційної роботи _____ Вікторія ОРЛЕНКО
Підпис Ініціали, прізвище

АНОТАЦІЯ

Тема кваліфікаційної роботи: «Системи виявлення вразливостей вебзастосунків».

Автор проекту: Кобець Яна Ростиславівна

Керівник проекту: Орленко Вікторія Сергіївна

Пояснювальна записка: 72 сторінки, 10 рисунків, 2 додатки, 33 джерела.

Графічна частина: 3 плакати, 16 презентаційних слайдів.

ВЕБЗАСТОСУНОК, СКАНУВАННЯ, СИСТЕМА ВИЯВЛЕННЯ ВРАЗЛИВОСТІ.

Метою даного проекту є дослідження та створення програмного забезпечення для виявлення вразливості вебзастосунків.

У даній кваліфікаційній роботі було проведено аналіз вебзастосунків, можливих ризиків нападу та витоку інформації, створення захисту та надійності програмного забезпечення для виявлення вразливості вебзастосунків, досліджено процес виявлення вразливостей веб-серверів та способи контролю цього процесу. Спроектовано та реалізовано структуру проекту.

Під час виконання проектування було досліджено предметну область, після чого реалізовано дієвий застосунок, який включає в себе всі можливі типи сканування для запобігання та виявлення як найбільшої кількості атак. Результатом дослідження є безперервне забезпечення безпеки та покращення процесів оцінки стану безпеки вебзастосунків.

1.06.2024

Дата

Підпис

ANNOTATION

Theme of the diploma project: "Web application vulnerability detection systems".

Author of the project: Kobets Yana Rostyslavivna

Project manager: Orlenko Viktoriia Serhiivna

Explanatory note: 72 pages, 10 figures, 2 appendixes, 33 sources.

Graphic part: 3 posters,16 presentation slides.

WEB APPLICATION, SCANNING, VULNERABILITY DETECTION SYSTEM.

The aim of this project is to research and create software for detecting vulnerabilities in web applications.

This thesis analyses web applications, possible risks of attack and information leakage, creates protection and reliability of software for detecting web application vulnerabilities, investigates the process of detecting web server vulnerabilities and ways to control this process. The project structure was designed and implemented.

During the design, the subject area was investigated, and then an effective application was implemented, which includes all possible types of scanning to prevent and detect as many attacks as possible. The result of the research is continuous security and improved processes for assessing the security of web applications.

1.06.2024

Date

Signature

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

DBS	–	Data Base Server
ООП	–	об’єктно-орієнтоване програмування
ПЗ	–	програмне забезпечення
БД	–	база даних
HTTPS	–	Hypertext Transfer Protocol Secure
HTML	–	HyperText Markup Language
JS	–	JavaScript
JWT	–	Json Web Tokens
Json	–	JavaScript Object Notation
CMS	–	Content Managment System
DOS	–	Denial-of-Service
SQL	–	Structured Query Language

ВСТУП

Основним завданням цієї кваліфікаційної роботи є створення функціонального додатку для виявлення вразливості вебзастосунків.

Сьогодні роль інтернету у сфері програмних продуктів значно зросла. Вебзастосунки, створені за допомогою веб-технологій, стали найкращим рішенням для багатьох завдань у різних галузях, поступово витісняючи додатки, що базуються на інших технологіях. Це призвело до ускладнення вебзастосунків у плані структури, архітектури та реалізації, а також до використання розподіленої архітектури. Такі зміни висунули нові вимоги до їх безпеки. У цій статті розглянуто найпоширеніші способи атак на вебзастосунки та методи захисту від зловмисників і недобросовісних користувачів.

Вразливість вебзастосунків залишається однією з найпоширеніших проблем у забезпеченні інформаційної безпеки. До інших частих проблем належать низький рівень обізнаності працівників з питань безпеки, слабка парольна політика або її невиконання, недоліки у процесах оновлення програмного забезпечення, використання небезпечних конфігурацій та неефективне міжмережеве розмежування доступу. Незважаючи на те, що вразливості вебзастосунків часто описані в науковій і спеціалізованій літературі, превентивні захисні механізми, які знижують ризики атак, зустрічаються рідко.

Постановка проблеми. Захищеність вебзастосунків ускладнюється тим, що при їх розробці часто не враховуються питання захисту від внутрішніх і зовнішніх загроз, або приділяється недостатньо уваги цьому процесу. Це призводить до ситуації, коли проблеми інформаційної безпеки стають очевидними для власника системи лише після завершення проєкту, а усунення вразливостей у вже створеному вебзастосунків є більш затратним, ніж на етапі його розробки та впровадження. Недооцінка серйозності ризику загроз інформаційній безпеці через вебзастосунки, доступні через Інтернет, є головним фактором низького рівня захисту.

Використання інформаційних систем і технологій пов'язане з певними ризиками. Оцінка ризиків необхідна для контролю ефективності діяльності у сфері інформаційної безпеки, прийняття відповідних захисних заходів та побудови ефективних систем захисту. Основою потенційних або реальних ризиків є можливі вразливості та загрози для національної безпеки, тому їх своєчасне виявлення, порушення конфіденційності та цілісності інформації, поширення шкідливого програмного забезпечення та фінансове шахрайство є головним завданням захисту вебзастосунків. Ризики потрібно постійно контролювати та періодично проводити їх переоцінку. Перша ретельно виконана та задокументована оцінка може значно спростити подальшу діяльність. Управління ризиками, як і будь-яку іншу діяльність у сфері інформаційної безпеки, необхідно інтегрувати у життєвий цикл інформаційної системи.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ФОРМУЛЮВАННЯ ЗАДАЧІ

1.1 Змістовний аналіз предметної області, її структурних та функціональних характеристик

Жага людини до науки та пізнаття чогось нового, спонукала до великих звершень в історії людства. За останнє 20-ти ліття наука невимовно стрімко пішла у гору. Всім нам за кожним новим винаходом здається, що це вже межа людської можливості, але як показує час та досвід з кожним роком в наше життя вриваються нові гаджети та винаходити.

На сьогодні, ми всі користуємося всесвітньою павутиною інформації Інтернет. Не виходячи з дому, не читаючи 10 книжок, відповідь майже на кожне наше запитання ми можемо віднайти у Інтернеті.

Батьком першого у світі веб-сайту став фізик Тім Бернерс-Лі [1]. Працюючи в Європейській організації ядерних досліджень (CERN) у Женеві, він у березні 1989 року запропонував використовувати гіпертекст для передачі даних через глобальну мережу Інтернет. У 1990 році він створив перший веб-сайт з адресою info.cern.ch, де детально описав нову технологію WWW (World Wide Web). Ця технологія базувалася на принципах інтернет-адресації URL, протоколу передачі даних HTTP та мові розмітки гіпертексту HTML.

Ще задовго до створення першого сайту, у 1980 році, Тім Бернерс-Лі розробив систему Enquire, яка мала на меті створити єдиний інформаційний простір, що включав Інтернет, комп’ютери користувачів та гіпертекст. Ця система дозволяла зберігати інформацію за принципом випадкових асоціацій, а гіпертекст забезпечував можливість не тільки переглядати інформацію на веб-сторінках, але й пов’язувати їх між собою за допомогою посилань, що забезпечувало легкий і швидкий доступ до даних. Для демонстрації нової технології, Бернерс-Лі запропонував співробітникам CERN розміщувати файли з гіпертекстом, пов’язані гіперпосиланнями, що дозволило працівникам отримувати доступ не лише до нових можливостей Глобальної мережі, але й до внутрішнього пошуку по документах.

На той час info.cern.ch був першим веб-сервером на базі ПК NeXT і першим у світі сайтом. На першій веб-сторінці містилася інформація про проект першого інтернет-браузера WWW, технічні особливості створення інших веб-сторінок та дані про гіпертекст. Там також були детальні пояснення про те, як здійснювати пошук необхідної інформації в Інтернеті, а також інструкції з інсталяції браузерів і серверів та роботи з ними. Перший у світі сайт став повноцінним інтернет-каталогом, де Бернерс-Лі пізніше публікував перелік посилань на нові сайти.

Сьогодні Тім Бернерс-Лі очолює власний Консорціум Всесвітньої Павутини (WWW Consortium), який спеціалізується на розробці та впровадженні нових інтернет-стандартів.

На сьогоднішній день вебзастосунками рахуються інтерактивні програми, які складаються з двох частин:

Одна частина завантажується у браузер, коли користувач вводить відповідну команду.

Інша частина розміщується на веб-сервері, де зберігаються всі дані, необхідні для роботи додатка. Веб-сервер отримує запит від користувача, знаходить потрібну інформацію та відправляє її назад у браузер. Для більшості користувачів вебзастосунки виглядають так само, як і звичайні сайти. Однак, якщо сайти переважно мають інформаційний характер, то вебзастосунки забезпечують широкий спектр функцій – від роботи з графікою та таблицями до оформлення замовлень онлайн.

Будь-які вхідні дані сайту організацій (теги введення, рядки запиту, файли cookie тощо) можуть бути використані для впровадження шкідливого коду.

Існують такі типи помилок SQL-ін'єкцій:

- неправильна обробка введення;
- помилка конфігурації СУБД на сервері;
- зміна умов запиту;
- часова затримка.

Неправильна обробка введення, такі помилки виникають, коли програміст або користувач неточно визначає введені дані або не виконує перевірку та

фільтрацію типів вхідних даних. Наприклад, якщо числове поле використовується в SQL-запиті, але програміст не враховує можливість введення некоректних типів даних.

Помилка конфігурації СУБД на сервері, уразливості можуть бути присутні в програмному забезпеченні баз даних сервера, як у випадку з функцією mysql_real_escape_string() на серверах MySQL. Це дозволяє зловмисникам здійснювати SQL-ін'єкцію за допомогою специфічних символів Юнікоду, навіть якщо введення проходить певну обробку.

Зміна умов запиту, така помилка дає змогу зловмиснику змінити умови в запиті, що призводить до некоректного відображення даних додатка.

Часова затримка, цей тип помилки виникає, коли час обробки одного або декількох SQL-запитів залежить від введених даних або коли обробка запитів займає значний час. Зловмисники можуть використовувати цей тип SQL-ін'єкції, щоб визначити точний час завантаження сторінки, коли введене значення правильне.

Помилки SQL-ін'єкцій (рис.1.1) виникають через ненадійне програмування, тому програмістам потрібно бути дуже уважними при розробці вебзастосунків.

ІТ-відділи на робочому місці мають засоби захисту конфіденційних даних, зокрема безпечні мережі, антивірусне програмне забезпечення, спеціальні брандмауери, онлайн-інструменти резервного копіювання тощо. Але, наприклад, працюючи вдома або в місцевій кав’ярні, працівники часто використовують незахищену мережу з іншими без належних засобів захисту. Це створює відкрите середовище для зловмисних атак, таких як фішинг і програми-вимагачі.

Середня вартість витоку даних зросла на 1,07 мільйона доларів через нещодавній перехід до глобального середовища віддаленої роботи. Крім того, результати показують, що 95% атак на кібербезпеку є результатом людської помилки.

На рисунку 1.1 зображено загальну діаграму SQL-ін’єкцій.

Рисунок 1.1 – Діаграма SQL-ін'єкцій

Коли співробітники працюють вдома, для хакерів доступно більше векторів атак, таких як телефони, планшети та ПК. Ще більш страшним є той факт, що менше 3% організацій захищають мобільні пристрої своїх співробітників.

Надання своїм співробітникам належного навчання з кібербезпеки WFH і WFA зменшить ймовірність успішної атаки соціальної інженерії або витоку даних, що вплине на вашу організацію. Оскільки багато співробітників мають доступ до конфіденційних даних, варто витрачати час на те, щоб інвестувати у своїх працівників так само, як і в оновлення програмного забезпечення та технологій.

WAF допомагає убезпечити вебзастосунки, фільтруючи і контролюючи HTTP-трафік між вебзастосунком та інтернетом. Працюючи на 7 рівні моделі OSI, відомому як "прикладний" рівень, WAF не забезпечує захист від усіх типів атак. Такий підхід до "пом'якшення" атак зазвичай включається в комплексний набір інструментів для повного захисту.

WAF зазвичай встановлюється перед вебзастосунком, виконуючи роль бар'єру між додатком та інтернетом. Якщо проксі захищає дані клієнта, то WAF діє як "зворотний проксі", захищаючи сервер і пропускаючи клієнтів через себе перед наданням доступу до ресурсу.

WAF функціонує на основі певного набору правил, спрямованих на захист додатків шляхом фільтрації шкідливого трафіку. Його перевага полягає у швидкій і простій модифікації цих правил, що дає змогу оперативно реагувати на нові вектори атак.

Як уже згадувалося, WAF працює на прикладному рівні, обробляючи потік даних, а не мережевий потік, після того як його отримав сервер. Це означає, що його зазвичай використовують після дешифрування, надаючи йому повний доступ до вмісту запитів і відповідей.

Існують сотні рішень WAF, кожне з яких має свої особливості. Деякі з них налаштовуються просто через веб-інтерфейс, де можна вмикати або вимикати фільтрацію певних категорій загроз. Інші потребують редагування текстових файлів, в яких небезпечні запити описуються на власній мові WAF. Для найсуворішого налаштування правил WAF необхідно активувати такі правила, які блокуватимуть усе, окрім необхідних типів запитів до веб-сервера. Для цього потрібно глибоко розуміти роботу захищеного веб-сайту, знати, які запити йому потрібні, а які ні, а також мати знання про типи атак, методи їх виконання і принципи роботи налаштовуваного WAF. Це складний процес, і такий ідеальний варіант трапляється нечасто.

1.2 Огляд існуючого програмного забезпечення предметної області

Із збільшенням розвитку та розробки вебзастосунків, разом із цим збільшився ризик витоку конфіденційної інформації. Тому що більшість веб-додатків так чи інакше зберають особисті дані своїх користувачів.Для початку потрібно розібратися як саме працюють вебзастосунки, а саме: вебзастосункок отримує запит від клієнта, виконує необхідні обчислення, створює веб-сторінку та відправляє її клієнту через мережу за допомогою протоколу HTTP. Вебзастосункок також може виступати клієнтом інших сервісів, таких як бази даних або інші вебдодатки, розміщені на різних серверах. У більшості вебдодатків

потрібно пройти авторизацію для того, щоб ними користуватися. Враховуючи, все те вище сказане, для того щоб все ж таки так чи інакше забезпечити свою особисту інформацію, створили система виявлення вразливостей вебзастосунків. Для кращого розуміння роботи та функціональної частини таких систем, хочу розглянути на прикладі роботи одного із таких інструментів Burp Suite [2].

Це потужний інструмент для тестування вразливостей вебдодатків, що допомагає забезпечити високий рівень безпеки вашого вебдодатку. Цей програмний комплекс містить набір інструментів для виявлення та експлуатації різноманітних типів вразливостей. Burp Suite дозволяє здійснювати аудит безпеки вебдодатків, перехоплювати та аналізувати трафік між клієнтом і сервером, перевіряти наявність вразливостей, таких як SQL-ін'єкції, XSS-атаки, CSRF-атаки, а також проводити тестування на проникнення.

XSS (міжсайтовий скриптинг) [20] є одним із видів атак на веб-системи, що полягає у впровадженні шкідливого коду на певну веб-сторінку. Цей код взаємодіє з віддаленим сервером зловмисників, коли користувач відкриває сторінку.

Термін Cross-Site Scripting скорочено як XSS, щоб уникнути плутанини з CSS (каскадні таблиці стилів.

Основна мета міжсайтового скриптингу – це викрадення файлів cookie користувачів за допомогою вбудованого скрипту на сервері, який потім використовується для отримання необхідних даних і проведення подальших атак. Зловмисник здійснює атаку не напряму на користувачів, а через уразливості веб-сайту, який вони відвідують, впроваджуючи спеціальний JavaScript. У браузері цей код виглядає як частина сайту, роблячи відвідуваний ресурс співучасником XSS-атаки.

Не існує чіткої класифікації міжсайтового скриптингу, але експерти по всьому світу виділяють три основні типи.

Збережений XSS (постійний). Це один із найнебезпечніших видів уразливостей, оскільки дозволяє зловмиснику отримати доступ до сервера і керувати шкідливим кодом (видаляти, модифікувати). Кожного разу при

зверненні до сайту виконується попередньо завантажений код, який працює автоматично. Найчастіше такі вразливості зустрічаються на форумах, порталах, блогах, де є можливість коментування в HTML без обмежень. Шкідливі скрипти можуть бути вбудовані як у текст, так і в зображення.

Cross-Site Request Forgery (CSRF) — це атака, яка змушує аутентифікованих користувачів надіслати запит до веб-застосунку, в якому вони зараз аутентифіковані [21]. CSRF-атаки використовують довіру вебзастосунку до аутентифікованого користувача. На відміну від атак міжсайтового скриптингу (XSS), які експлуатують довіру користувача до певного веб-застосунку, CSRF-атаки використовують уразливість веб-застосунку, якщо він не може розрізнити запит, створений індивідуальним користувачем, та запит, створений без згоди цього користувача.

Щоб запобігти атаці CSRF, додатки повинні мати спосіб визначати, чи HTTP-запит був законно створений через інтерфейс користувача додатка. Найефективніший спосіб досягти цього — використання CSRF-токена. CSRF-токен — це безпечний випадковий токен (наприклад, синхронізуючий токен або токен виклику), який використовується для запобігання атакам CSRF. Токен має бути унікальним для кожної користувацької сесії та мати великий випадковий значення, щоб його було важко вгадати.

CSRF-захищений додаток присвоює унікальний CSRF-токен для кожної користувацької сесії. Ці токени вставляються у приховані параметри HTML-форм, пов'язаних із критичними серверними операціями, і надсилаються в браузери клієнтів.

Відповідальність за визначення чутливих серверних операцій лежить на команді розробників додатка. CSRF-токени повинні бути частиною HTML-форм і не зберігатися в сесійних cookie. Найпростіший спосіб додати непередбачуваний параметр — використати безпечну хеш-функцію (наприклад, SHA-2) для хешування ідентифікатора сесії користувача. Для забезпечення випадковості токени повинні генеруватися криптографічно безпечним генератором випадкових чисел.

Коли користувач виконує ці критичні операції, запит, згенерований браузером, повинен включати відповідний CSRF-токен. Сервер додатка використовує цей токен для перевірки законності запиту від кінцевого користувача. Якщо CSRF-токен не відповідає, сервер відхиляє запит.

Burp Suite забезпечує як ручне, так і автоматичне тестування, що дозволяє максимально ефективно виявляти вразливості. Крім того, цей інструмент дозволяє налаштовувати спеціальні скрипти для автоматизації процесу тестування та виявлення вразливостей. Завдяки своїм можливостям Burp Suite є незамінним для команд розробників та тестувальників, які займаються створенням і тестуванням вебдодатків. Використання цього інструменту дозволяє забезпечити високий рівень безпеки вебдодатків і звести ризик їх вразливості до мінімуму.

Якщо ви займаєтеся розробкою або тестуванням вебдодатків, Burp Suite – це інструмент, який обов'язково варто мати. Він допоможе вам ефективно тестувати вразливості та забезпечити високий рівень безпеки вашого вебдодатку.

Сьогодні ми розглянемо основні інструменти, які Burp Suite використовує для покращення безпеки. В Burp Suite є кілька важливих вкладок, серед яких найпопулярніші: Proxy, Intruder, Repeater, Decoder, Logger і Comparer. Ми також ознайомимося з OWASP Top 10 і OWASP Juice Shop. Дізнаємося про найпоширеніші категорії вразливостей і на практиці спробуємо експлуатувати одну з них за допомогою Juice Shop, вебдодатку для розвитку навичок тестування на вразливості.

Не менш важливий є організаційний аспект при створенні системи захисту вебдодатків — це тест на проникнення. Він є оптимальним способом перевірки безпеки інформаційної системи шляхом імітації цілеспрямованих атак. Тест на проникнення дозволяє оцінити захищеність системи від несанкціонованих впливів, використовуючи різні моделі вторгнень.

Тестування на проникнення для вебдодатків зосереджується виключно на оцінці рівня захисту вебдодатків. Процес включає активний аналіз додатків і

пошук вразливостей, технічних помилок та інших проблем. Усі виявлені слабкі місця фіксуються в підсумковому звіті.

Тобто, враховуючи те що в наш час вебдодатки удосконаленні, та в більшій мірі є безпечними для використання, не потрібно забувати, що все ж таки може знайтися слабке місце для витоку інформації і щоб уникнути цього використовуються різні системи захисту згідно до уподобань користувача. Потрібно також пам’ятати, що завантаження таких систем потрібно проводити лише з офіційних сайтів та перевірених постачальників, потрібно також мати розуміння того, що кожен продукт має свою вартість і хоч в наш час є велика кількість піратських сайтів з яких можна завантажити, що завгодно, потрібно пам’ятати про те, що це може призвести до фатальних наслідків для вашого комп’ютера та інформації, яка на нього знаходиться іноді.

Традиційний WAF фільтрує окремі запити, наприклад, блокуючи ті, що містять подвійне кодування, спроби впровадження коду або надходять з IP-адрес, позначених як ворожі. Багато з цих загроз включені в топ-10 ризиків безпеки веб-застосунків OWASP.

Nxgen WAF виконує всі ці функції та багато іншого. Він аналізує не лише зміст і формат кожного запиту, але й відстежує поведінку запитувача в ширшому контексті. Наприклад, атаки типу DoS або DDoS складаються з великої кількості запитів, які окремо можуть виглядати безпечними. Сучасний WAF робить більше, ніж просто фільтрує трафік на основі окремих запитів; він аналізує загальну активність кожного запитувача, враховуючи весь створений трафік, щоб визначити його наміри (легітимні чи зловмисні). Це дозволяє захищати від широкого спектру веб-загроз.

Програмні WAF та апаратні пристрої є складними продуктами, які стають ще складнішими зі зростанням інтернет-загроз. Це ускладнює їх адміністрування та управління, а також вимагає частих виправлень і оновлень.

Для WAF наступного покоління ці проблеми вже не є актуальними. Провайдер може дистанційно керувати та оновлювати хмарні WAF, що значно полегшує їхню експлуатацію.

Традиційні WAF використовують два основні методи для ідентифікації шкідливого трафіку: сигнатурне виявлення та геолокацію. Сигнатурне виявлення перевіряє вхідні запити, порівнюючи їх з базою даних загроз. Якщо запит відповідає відомому шаблону загрози, він блокується; якщо ні, то дозволяється.

Геолокаційна фільтрація блокує запити, що надходять з джерел, які вважаються зловмисними. База даних шкідливих джерел зазвичай складається зі списків сторонніх розробників (наприклад, Spamhaus EDROP), а також зі статичних та динамічних внутрішніх списків.

Хоча ці методи були ефективними протягом багатьох років, сучасні загрози навчилися обходити їх. Запити часто не відповідають відомим сигнатурам загроз, а деяка зловмисна вебактивність (наприклад, сканування сайту) використовує цілком легітимні запити. Геолокаційну фільтрацію можна обійти, змінюючи IP-адреси, що зловмисники часто роблять за допомогою стільникових шлюзів та інших методів доступу до великої кількості IP-адрес.

WAF наступного покоління вирішують ці проблеми, не покладаючись виключно на сигнатурне виявлення та геолокацію. Хоча ці методи залишаються корисними для швидкої ідентифікації та блокування простіших атак, сучасні WAF також використовують аналіз поведінки запитувача для виявлення загроз.

Аналіз поведінки дозволяє WAF відстежувати джерела трафіку навіть при зміні IP-адрес або клієнтських пристроїв. Сучасні WAF, як-от AWS WAF, застосовують машинне навчання та інші технології для створення складних поведінкових профілів легітимних користувачів. Під час кожної взаємодії з вебзастосунком аналізуються та порівнюються статистика пристрою, події інтерфейсу користувача, хронометраж та показники сесії. Ворожі користувачі зрештою відхиляються від легітимної поведінки, і коли це трапляється, WAF блокує їхній подальший доступ до мережі.

Традиційні WAF були статичними застосунками або пристроями, які вимагали ручного налаштування та адміністрування. Це робить їх непридатними для сучасних практик, таких як DevOps, DevSecOps, незмінна інфраструктура тощо.

WAF наступного покоління, такі як AWS WAF, працюють у хмарі, що дозволяє їх розгортати, налаштовувати та керувати ними програмно. Вони легко інтегруються в робочі процеси CI/CD, а також підтримують безсерверну та контейнерну архітектури. Такі WAF є динамічними та адаптивними, вони можуть розпізнавати зміни в трафіку (наприклад, нові виклики до API) і пропонувати відповідні зміни в наборі правил для адміністраторів.

1.3 Встановлення вимог до програмного забезпечення та створення ТЗ

Веб-застосунки мають специфіку, яка дозволяє в одному сеансі користувача з веб-сервером відкривати багато різних TCP-з'єднань з різних адрес, але з одним ідентифікатором сесії (можливо, динамічним). Атаки на веб-ресурси, які маніпулюють програмним забезпеченням з метою досягнення шкідливих цілей, зазвичай відбуваються одночасно з сесіями легітимних користувачів і використовують стандартні HTTP-порти. Блокування всього трафіку на цих портах недоцільне, оскільки це призведе до повного закриття доступу до веб-ресурсів як ззовні, так і зсередини. Ця проблема мережевої безпеки є чудовою можливістю для хакерів шукати вразливості на рівні вебзастосунків. Рішенням цієї проблеми є використання брандмауерів вебзастосунків (WAF).

Ринок WAF-брандмауерів є відносно новим, але великі гравці, такі як Amazon (AWS WAF), Citrix (NetScaler Web App Security Service), Cloudflare (Cloudflare WAF Service), Oracle (Oracle WAF) та Microsoft (Microsoft Azure WAF) вже пропонують готові рішення. Однак, за даними компанії Gartner, лідерами в цьому сегменті є рішення від компаній Akamai та Imperva.

WAF можна розгорнути у вигляді мережевих, автономних або хмарних рішень. Мережеві WAF зазвичай представлені у вигляді апаратних пристроїв, що поєднують функції класичних мережевих фільтрів і файєрволів вебзастосунків. Автономні WAF зазвичай є програмними комплексами, які встановлюються як настільні додатки на комп'ютерах-хостах. Хмарні WAF виконуються як мережеві

сервіси та надаються користувачам у вигляді послуг типу SaaS (software as a service).

Основні функції WAF включають:

- фільтрацію мережевих протоколів;
- фільтрацію протоколів HTTP;
- моніторинг стану з'єднань;
- підтримку високої доступності;
- контроль управління сеансами: оцінка потоків трафіку, використання тайм-аутів, моніторинг перехоплень сеансів;
- моніторинг і захист файлів cookie;
- контроль за значеннями прихованих полів;
- моніторинг перебору значень (захист від brute-force атак).

Основні вимоги система виявлення вразливостей вебзастосунків можна розділити на два основні пункти, це:

- функціональні;
- нефункціональні.

Для кращого розуміння та розробки даного програмного забезпечення розберемося що відноситься для початку до функціональних вимог:

- аудит безпеки — можливість ручного та автоматизованого тестування;
- перехоплення та аналіз трафіку — перехоплення HTTP/HTTPS трафіку між клієнтом та сервером та аналіз і збереження логів для подальшого дослідження;
- сканування на вразливість — виявлення поширених вразливостей, таких як SQL-ін’єкції, XSS, CSRF тощо та можливість налаштування глибини та детальності сканування;
- автоматизація тестування — налаштування та запуск автоматизованих тестів. Підтримка сценаріїв автоматизації для різних типів вразливостей;
- інтергація з іншими системами — підтримка інтеграції з CI/CD системами для автоматизованого тестування та можливість взаємодії з системами управління інцидентами (наприклад, Jira, ServiceNow);

- керування користувачами — адміністрування користувачів, ролей та прав доступу, а також аутентифікація та авторизація користувачів;
- налаштування системи — гнучке налаштування параметрів системи та сканування, плюс до цього ще, управління конфігураціями та оновленнями системи.

До нефункціональних вимог виключено п’ять основних:

- продуктивність — швидке та ефективне виявлення вразливостей без значного впливу на продуктивність везастосунків;
- безпека — захист даних під час тестування та зберігання звітів;
- надійність системи — висока доступність та стійкість до збоїв;
- масштабованість — обробка великої кількості вебзастосунків одночасно;
- юзабіліті — інтуїтивно зрозумілий інтерфейс для адмінів та тестувальників, а також навчальний матеріал для користувачів системи.

Нефункціональна вимога (NFR) описує якісний атрибут програмного забезпечення. Вона оцінює систему за параметрами, такими як швидкість відгуку, зручність використання, безпека, портативність та іншими нефункціональними критеріями, які є критично важливими для успіху програмного забезпечення. Наприклад, однією з нефункціональних вимог є час завантаження веб-сайту. Якщо нефункціональні вимоги не будуть виконані, система може не задовольнити потреби користувачів.

Функціональна вимога описує поведінку системи і визначає, які дії вона повинна виконувати для задоволення потреб або очікувань користувачів. Функціональні вимоги можна розглядати як функціональні можливості, доступні користувачу. Вони відрізняються від нефункціональних вимог, що встановлюють внутрішні параметри роботи системи, такі як продуктивність, безпека тощо.

Тому при проектуванні структури проекту, важливо включити всі ці вимоги. Також функціональні та нефункціональні вимоги діляться на підпункти, згідно яких потім і буде розроблено дієву систему виявлення вразливості вебзастосунків.

На рисунку 1.2 відображено схему основних вимог система виявлення вразливостей вебзастосунків.

Рисунок 1.2 – схема основних вимог системи виявлення вразливості

Дана схема сконструктована для того, щоб полегшити розуміння під час розробки застосунку, який функціонал повинен бути в ньому.

2 ПРОЕКТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1 Ахітектура та функціональні характеристики додатка

Проектуючи будь-який додаток, кожен розробник повинен розробити свою особисту архітектуру на яку впливає велика кількість факторів. Для початку потрібно розібратися із призначенням додатку, вибрати технологію за якою буде розроблятися даний додаток, створити схему за допомогою якої в подальшому реалізовувати технічний та логічний функціонал ПЗ.

Згідно теми даної кваліфікаційної роботи основною задачею є створення система виявлення вразливостей вебзастосунків . Архітектура система виявлення вразливостей вебзастосунків може бути складною та багаторівневою, адаптованою під потреби забезпечення безпеки та ефективності сканування. Приклад типової архітектури такої системи:

- інтерфейс користувача (User Interface, UI) [3];
- серверна логіка (Server-Side Logic);
- база даних (Database);
- модулі сканування (Scanning Modules) [22];
- інтеграція та API.

Веб-інтерфейс забезпечує доступ користувачів до системи через браузер, дозволяє налаштувати параметри сканування, запускати сканування, переглядати результати та управляти конфігурацією.

Консоль командного рядка (CLI) дозволяє більш досвідченим користувачам та автоматизаційним системам використовувати функціонал системи через консольні команди. Інтерфейс повинен бути зрозумілим та дотупним для розуміння користувачів.

Модуль управління скануванням керує запуском і контролем процесів сканування, розподіляє завдання між агентами сканування. Модуль обробки результатів збирає, аналізує та обробляє дані, отримані від агентів сканування. Генерує звіти, визначає рівні ризику та класифікує вразливості.

База даних вразливостей містить інформацію про відомі вразливості, патчі, оновлення та методи атак. Сховище результатів сканування зберігає детальні результати сканувань, історію сканувань і їхні результати.

Статичний аналізатор проводить статичний аналіз коду додатків на предмет виявлення вразливостей на етапі розробки. Динамічний аналізатор виконує динамічне тестування вебдодатків, імітуючи атаки на живій системі для виявлення потенційних вразливостей. Аналізатор залежностей перевіряє бібліотеки та залежності, які використовуються в додатку, на предмет відомих вразливостей. На рисунку 2.1. зображено сканування вебдодатка, тобто безпосередню роботу системи виявлення вразливості.

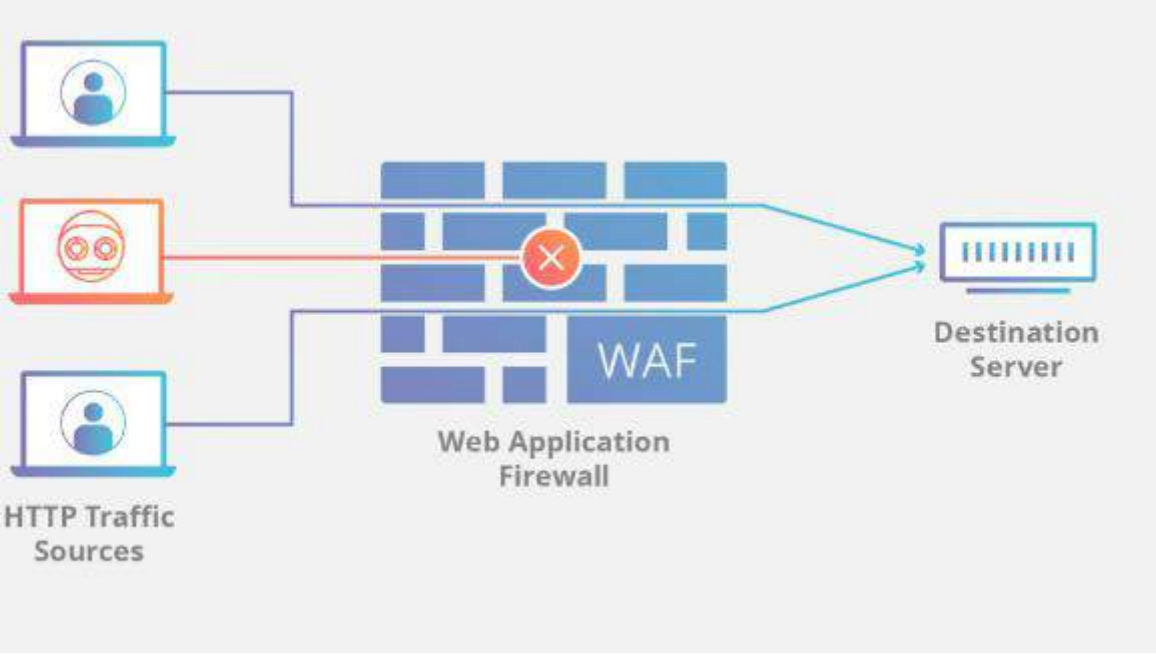


Рисунок 2.1 – Сканування вебзастосунка через WAF

API забезпечує інтеграцію з іншими системами, дозволяючи автоматизувати процеси, інтегрувати результати сканування з іншими інструментами безпеки або системами розробки. Плагіни та інтеграції дозволяють розширювати функціональність системи, інтегруватися з зовнішніми сервісами та інструментами, наприклад, з системами CI/CD, системами управління версіями. Функціональні характеристики додатка системи виявлення вразливостей

вебдодатків описують ключові можливості та дії, які система може виконувати для ефективного забезпечення безпеки вебдодатків. Схема безпосередньої роботи такої системи зображено на рисунку 2.2. До основних функцій відноситься рохширене виявлення вразливості, тобто виявлення широкого спектру вразливостей, а саме система здатна ідентифікувати різні типи вразливостей, включаючи SQL ін'єкції, XSS, CSRF, переповнення буфера, неправильні конфігурації безпеки тощо. На рисунку 2.3. наведено приклад атак на серверні додатки. У випадку SQL ін'єкції, суть полягає у впровадженні випадкового SQL-коду в дані, що передаються через запити GET або POST. Якщо веб-ресурс приймає та виконує такий код, це означає, що він є вразливим, і з його базою даних можна взаємодіяти без обмежень.

Рисунок 2.2 – Схема роботи WAF

Не менш важливим є керування та моніторинг, тобто контроль доступу - налаштування прав доступу для різних користувачів або груп, забезпечення сегрегації обов'язків. Моніторинг стану системи - засоби для моніторингу стану сканувань та виявлення можливих проблем в роботі системи.

Міжмережевий екран для вебдодатків (WAF) використовує дві загальноприйняті моделі NEGATIVE та POSITIVE. Negative - це негативна модель або чорний список, яка заперечує все, що є свідомо встановленим. WAF може використовувати загальновідомі сигнатури для запобігання найпоширенішим атакам, а також специфічні сигнатури для атак, які використовують уразливості конкретних вебдодатків.

Рисунок 2.3 – Приклад атак на вебзастосунок

Наприклад, може бути встановлено правило, що відхиляє певні потенційно небезпечні HTTP-запити[31] GET і дозволяє все інше. Positive - це позитивна модель або білий список, яка дозволяє тільки те, що є свідомо встановленим. Для поліпшення захисту, на додаток до сигнатур, використовуються правила, які визначають, що є явно дозволеним. Наприклад, може бути встановлено правило, яке дозволяє лише HTTP-запити GET для певного URL і забороняє все інше. Наростаючий тиск конкуренції на сучасному ринку різних галузей створив дуже чутливе до зовнішнього середовища веб-інфраструктуру. Сучасні архітектури розгортання вебдодатків є складними та вимагають інтеграції багатьох різнорідних технологій, що створює потенціал для численних вразливостей. Швидкі цикли розробки та постійні оновлення вебдодатків подальше погіршують ситуацію. У таких умовах ризики для бізнесу компаній стають надзвичайно великими, оскільки вразливості в вебдодатках ставлять під загрозу критично важливі бізнес-процеси та конфіденційні дані. Значні фінансові збитки можуть виникнути в результаті непередбачених затримок у бізнес-операціях, крадіжки інтелектуальної власності, втрати довіри клієнтів і порушення репутації бренду. У багатьох випадках безпека вебдодатків є юридичною вимогою, наприклад, дотримання стандарту безпеки даних із збереження платіжних карт (Payment Card Industry Data Security Standard - PCI DSS). Сучасні тенденції розвитку веб-сервісів свідчать про відсутність загальноприйнятих стандартів безпечного

програмування вебдодатків, що призводить до помилок у розробці програмного забезпечення та виникнення серйозних вразливостей у веб-сервісах, які використовують такі додатки. Ускладнення становища полягає в тому, що вразливий вебдодаток може бути легко скомпрометований, навіть без використання спеціалізованих засобів, лише за допомогою веб-браузера. Захист від вразливих вебдодатків можна забезпечити шляхом виправлення вразливостей у вебдодатку або застосування спеціалізованих засобів захисту вебдодатків, таких як брандмауери для веб-застосунків (Web Application Firewall - WAF). У сучасних умовах розташування традиційних брандмауерів, брандмауерів нового покоління (Next Generation Firewalls - NGFW) або систем запобігання вторгненням (Intrusion Prevention Systems - IPS)[32] на периметрі мережі або як шлюзів для довірених сегментів мережі є недостатнім заходом для забезпечення повного захисту мережевої інфраструктури. Атаки на веб-ресурси, які використовують програмне забезпечення для досягнення зловмисних цілей, як правило, відбуваються одночасно з сесіями легітимних користувачів і, переважно, використовують стандартні HTTP (80) і HTTPS (443) порти. Блокування всього трафіку на рівні портів не є ефективним виходом з цієї ситуації, оскільки це призведе до повного закриття доступу до вебдодатків як ззовні, так і зсередини. Така дилема в мережевій безпеці створює ідеальні умови для пошуку хакерами вразливостей на рівні вебдодатків.

2.2 Опис структури даних та моделі бази даних

База даних – це невід’ємна частина будь-якого програмного забезпечення [4]. За дапомогою бази даних. Перше з чого починається робота з БД це з її проектування [23]. База даних (БД) являє собою впорядковану сукупність взаємопов'язаних даних, організовану за певними правилами. Таблиці є основними компонентами баз даних, складаючись з рядків і стовпців для зберігання конкретного набору даних. У кожній таблиці кожен стовпець

представляє атрибут або характеристику, а рядки відображають записи або конкретні дані. Наприклад, у таблиці "Користувачі" можуть бути стовпці, такі як "Ім'я", "Прізвище", "Email" і "Дата реєстрації", а рядки відображатимуть інформацію про окремих користувачів. Взаємозв'язки між таблицями дають можливість зв'язувати дані з різних таблиць, сприяючи уникненню дублювання і забезпечуючи цілісність інформації. Вони можуть бути різних типів, таких як "один до багатьох", "багато до багатьох" і "один до одного". Наприклад, у базі даних інтернет-магазину може існувати відношення "один до багатьох" між таблицями "Покупці" і "Замовлення", де один покупець може мати кілька замовлень. Запити - це засіб отримання інформації з бази даних, що дозволяють вибирати, додавати, змінювати або видаляти дані, а також об'єднувати дані з різних таблиць. Для роботи з запитами використовуються мови запитів, такі як SQL (Structured Query Language). Наприклад, запит може мати такий вигляд:

SELECT[33] ім'я, прізвище FROM користувачі WHERE email = 'example@email.com';

У цьому прикладі ми вибираємо ім'я та прізвище користувача з певною електронною адресою.

Індекси використовуються [30]для прискорення пошуку даних у таблиці. Вони працюють на зразок каталогу книги, вказуючи на місцезнаходження даних для певного значення атрибута. Однак індекси можуть займати додаткове місце і уповільнювати операції вставки або оновлення даних, тому їх варто використовувати розумно. Структура бази даних — це офіційний опис у вигляді схеми, що визначає взаємозв'язки та структуру таблиць та інших об'єктів у базі даних. Вона включає визначення таблиць, стовпців, типів даних, обмежень, відносин та індексів. Структура допомагає забезпечити цілісність даних і спрощує роботу з базою даних для розробників і адміністраторів.

Зазвичай процес проектування баз даних починається з розробки моделі предметної області, для якої створюється БД. Розробка баз даних є ітеративним та багатоетапним процесом, що передбачає прийняття обґрунтованих рішень під час аналізу моделі даних, врахування вимог з боку розробників програмного

забезпечення та користувачів, а також створення логічних та фізичних структур даних, аналіз та вибір програмних та апаратних засобів.

Процес розробки баз даних включає чотири основні етапи:

- системний аналіз та опис інформаційних об'єктів;
- розробка інфологічної моделі або Концептуальне проектування;
- логічне проектування баз даних;
- фізичне проектування баз даних.

Аналіз системи передбачає опис реальних об'єктів предметної області, встановлення зв'язків між ними та вивчення характеристик об'єктів і зв'язків. Отримані результати використовуються для створення концептуального проекту бази даних.

Для визначення складу і структури предметної області застосовуються два підходи: функціональний і предметний.

Функціональний підхід виходить "від задач"[29] і використовується, коли відомі функції майбутніх користувачів бази даних і відомі всі задачі, для яких створюються БД. Це дозволяє чітко визначити мінімальний набір об'єктів предметної області та їх взаємозв'язок на основі виробничих документів і опитувань замовників.

Предметний підхід застосовується, коли інформаційні потреби майбутніх користувачів не визначені чітко. У цьому випадку в опис предметної області включаються об'єкти та зв'язки, які є найбільш характерними та суттєвими. БД, створена за цим підходом, може використовуватися для рішення завдань, які заздалегідь не були визначені.

У практичній діяльності використовується комплексний підхід, що дозволяє вирішувати конкретні завдання та враховувати можливість додавання нових застосувань. Існують два основні підходи до проектування БД: низхідне і висхідне проектування.

Низхідне проектування починається з визначення наборів даних, потім визначаються елементи даних для кожного з них. Цей процес включає ідентифікацію різних типів сутностей і визначення їх атрибутів.

Висхідне проектування починається з виявлення елементів даних, які потім групуються в набори. Спочатку визначаються атрибути, які потім об'єднуються в сутності. Висхідне проектування включає операції синтезу, що передбачає створення зазначених функціональних залежностей між об'єктами предметної області вихідних відношень схеми БД.

Починаючи з концептуального проектування, створюється концептуальна схема бази даних, що ґрунтується на концептуальній моделі даних.

Концептуальна модель[27] надає загальне уявлення про дані. В концептуальному проектуванні використовуються два основні підходи до моделювання даних:

- семантичні моделі;
- об'єктні моделі.

Які акцентують увагу на структурі даних. Однією з найбільш поширених семантичних моделей є модель "сутність - зв'язок" (ER-модель). ER-модель складається з сутностей, зв'язків, атрибутів, доменів атрибутів і ключів. Моделювання даних відображає логічну структуру інформації, аналогічно тому, як блок-схеми алгоритмів відображають логічну структуру програм.

Які зосереджені на поведінці об'єктів даних і методах їх обробки. Основним поняттям таких моделей є об'єкт, який має стан (атрибути) і поведінку (методи). Стан об'єкта визначається його атрибутами, а поведінка - набором операцій, доступних для цього об'єкта.

Ці дві моделі часто об'єднуються в розширеній ER-моделі (EER-модель), що дозволяє більш гнучке і точне моделювання структури даних і їх поведінки.

Логічне проектування включає створення моделі даних, що ґрунтується на обраній концептуальній моделі. На цьому етапі необхідно визначити, яка система управління базами даних (СУБД)[29] буде використовуватися в системі (наприклад, ієрархічна, мережева, реляційна чи об'єктно-орієнтована). Для перевірки правильності логічної моделі застосовується процес нормалізації. Крім того, логічна модель перевіряється на відповідність всім вимогам транзакцій користувачів.

Фізичне проектування охоплює опис ресурсів для фізичної реалізації логічного проекту бази даних. Фізичні моделі визначають розміщення даних в середовищі зберігання і засоби доступу до цих даних, які підтримуються на фізичному рівні. У процесі проектування визначається структура реляційної бази даних, включаючи склад таблиць, їх структуру і логічні зв'язки. Структура таблиці визначається за допомогою визначення стовпців, типів даних і розмірів стовпців, а також використанням ключів.

Фізичне проектування[26] включає в себе опис ресурсів, які використовуються для конкретної реалізації логічного проекту бази даних. Фізичні моделі визначають способи зберігання даних і методи доступу до цих даних на рівні фізичної реалізації.

У моделі БД визначаються типи об'єктів, що будуть включені до бази даних, та зв'язки між ними. Для створення БД для системи виявлення вразливості вебдодатка структура даних та модель повинні забезпечувати зберігання та ефективне керування інформацією про вразливості, сканування, користувачів та інші критичні компоненти. Логічна схема бази даних визначає структуру даних, взаємозв'язки між таблицями та їх атрибути. Нижче наведено логічну схему бази даних для системи виявлення вразливостей вебдодатків.

Першою та ключовою таблицею у БД є таблиця USERS. Вона зберігає всю необхідну інформацію про користувачів системи. Нижче наведено опис призначення кожного поля в таблиці.

Нижче подано опис таблиці Users:

user_id (PK) ― унікальний ідентифікатор користувача, який забезпечує унікальність кожного запису в таблиці. Використовується для зв'язку з іншими таблицями в базі даних;

username ― ім'я користувача, яке використовується для входу в систему. Це поле дозволяє користувачеві ідентифікувати себе в системі;

password_hash ― хеш пароля, що зберігається для аутентифікації користувача. Хешування пароля забезпечує безпеку, оскільки паролі зберігаються в захищеному вигляді;

email — електронна адреса користувача, яка використовується для зв'язку з користувачем, відновлення доступу та для інших комунікацій;

role — роль користувача в системі, наприклад, адміністратор, розробник, тестувальник. Це поле визначає рівень доступу та привілеї користувача, контролює доступ до різних функцій системи;

created_at — дата та час створення облікового запису користувача. Це поле дозволяє відстежувати, коли обліковий запис був створений;

updated_at — дата та час останнього оновлення даних користувача. Це поле допомагає відстежувати зміни, внесені до облікового запису, і підтримувати актуальність інформації.

Наступною за послідовністю є таблиця SCANS. Її призначення полягає у зберіганні інформації про всі проведенні сканування вебдодатків. Нижче наведено поля та їх опис:

scan_id (PK) — унікальний ідентифікатор сканування, який забезпечує унікальність кожного запису в таблиці. Використовується для посилань на конкретне сканування в інших таблицях;

user_id (FK) — ідентифікатор користувача, який запустив сканування. Це поле є зовнішнім ключем, що зв'язує запис зі скануванням з користувачем у таблиці Users;

scan_date — дата та час початку сканування. Це поле допомагає відстежувати, коли було розпочато конкретне сканування;

status — статус сканування (наприклад, запущено, завершено, помилка). Це поле дозволяє відстежувати поточний стан процесу сканування;

target_url — цільовий URL для сканування, тобто адреса вебдодатку або веб-сторінки, що сканується. Це поле визначає об'єкт сканування;

scan_type — тип сканування (наприклад, статичний, динамічний, аналіз залежностей). Це поле вказує на метод або техніку, що використовується для сканування;

created_at ― дата створення запису про сканування. Це поле дозволяє відстежувати, коли було створено запис про конкретне сканування;

updated_at ― дата останнього оновлення даних. Це поле допомагає відстежувати зміни, внесені до запису про сканування, і підтримувати актуальність інформації.

Задля того, щоб по скануванню інформація була збережена та оброблена програмою стовюю таблицю результату сканування SCANRESULTS. Нижче наведено назви та опис полів даної таблиці:

result_id (PK) ― унікальний ідентифікатор результату сканування, який забезпечує унікальність кожного запису в таблиці;

scan_id (FK) ― ідентифікатор сканування, до якого належить цей результат. Це поле є зовнішнім ключем, що зв'язує результат зі скануванням у таблиці Scans;

vulnerability_id (FK) ― ідентифікатор виявленої вразливості. Це поле є зовнішнім ключем, що зв'язує результат з конкретною вразливістю в таблиці Vulnerabilities;

severity ― рівень критичності вразливості (низький, середній, високий). Це поле допомагає класифікувати вразливості за їх важливістю;

description ― опис вразливості. Це поле містить детальну інформацію про виявлену вразливість, включаючи її природу та потенційні наслідки;

recommendations ― рекомендації щодо усунення вразливості. Це поле містить поради та заходи, які слід вжити для виправлення вразливості;

created_at ― дата створення запису про результат сканування. Це поле допомагає відстежувати, коли було створено запис про конкретний результат сканування;

updated_at ― дата останнього оновлення даних. Це поле дозволяє відстежувати зміни, внесені до запису про результат сканування, і підтримувати актуальність інформації.

Логічним буде після сканування та визначення вразливостей створити таблицю, яка буде запам’ятовувати інформацію про відомі вразливості, які можуть бути виявленні під час сканування.

Нижче наведено назви та опис полів таблиці Vulnerabilities:

vulnerability_id (PK) ― унікальний ідентифікатор вразливості, який забезпечує унікальність кожного запису в таблиці;

name ― назва вразливості, що дозволяє ідентифікувати її тип або категорію;

description ― опис вразливості, що містить детальну інформацію про природу вразливості, можливі наслідки її експлуатації та загальну інформацію про проблему;

cwe_id ― ідентифікатор загальної слабкості (Common Weakness Enumeration), який використовується для класифікації типу вразливості. Це стандартний ідентифікатор, що полегшує інтеграцію з іншими системами безпеки та довідниками;

cvss_score ― оцінка CVSS (Common Vulnerability Scoring System), що визначає рівень критичності вразливості на основі її впливу та можливості експлуатації. Ця оцінка допомагає пріоритизувати вразливості за ступенем їхньої загрози;

created_at ― дата створення запису про вразливість. Це поле допомагає відстежувати, коли було додано інформацію про конкретну вразливість;

updated_at ― дата останнього оновлення даних. Це поле дозволяє відстежувати зміни, внесені до запису про вразливість, і підтримувати актуальність інформації.

Задля того, щоб система мала збережену інформацію про конфігураціїні параметри, яка використовується під час сканування вебдодатка створюється наступна таблиця SCANCONFIGURATION.

Нижче наведено назви та опис полів таблиці ScanConfigurations:

config_id (PK) ― унікальний ідентифікатор конфігурації, який забезпечує унікальність кожного запису в таблиці;

scan_id (FK) — ідентифікатор сканування, що зв'язує конфігураційні параметри з конкретним скануванням. Це поле служить зовнішнім ключем, посилаючись на запис у таблиці Scans;

parameter_name — назва параметра конфігурації, що визначає, який саме аспект сканування налаштовується. Це може бути параметр, такий як глибина сканування, тайм-аут запиту, тощо;

parameter_value — значення параметра конфігурації, що встановлює конкретне налаштування для параметра. Наприклад, якщо параметр називається "timeout", значенням може бути "30 секунд";

created_at — дата створення запису про конфігурацію. Це поле допомагає відстежувати, коли було додано конфігураційний параметр до бази даних;

updated_at — дата останнього оновлення даних. Це поле дозволяє відстежувати зміни, внесені до конфігураційного параметра, і підтримувати актуальність налаштувань.

Таблиця ScanConfigurations є важливим компонентом для гнучкого та точного налаштування процесу сканування, забезпечуючи можливість детального контролю параметрів і оптимізації процесу виявлення вразливостей.

Щоб зберігати інформацію про різні події, що відбуваються в системі виявлення вразливості вебдодатка створюю наступну таблицю EVENTLOGS.

Нижче наведено назви та опис полів таблиці EventLogs**:**

event_id (PK) — унікальний ідентифікатор події, який забезпечує унікальність кожного запису в таблиці;

user_id (FK) — ідентифікатор користувача, який ініціював подію. Це поле служить зовнішнім ключем, посилаючись на запис у таблиці Users;

event_type — тип події, що відображає характер дії, здійсненої в системі. Наприклад, це може бути вхід в систему, запуск сканування, оновлення налаштувань тощо;

event_description — опис події, що надає детальну інформацію про те, що саме відбулося. Це може включати додаткові деталі, такі як параметри сканування або налаштування, що були змінені;

event_date — дата та час події, що дозволяє точно визначити момент, коли подія відбулася. Це важливо для відстеження та аудиту дій користувачів в системі.

Дана таблиця є важливим компонентом системи для ведення журналу подій, забезпечуючи можливість відстеження та аналізу дій користувачів, а також підтримки безпеки та прозорості в системі виявлення вразливостей. Після проведення аналізу та розробки проекту бази даних, була створена логічна структура бази даних, яка відображена на діаграмі рисунок 2.4.

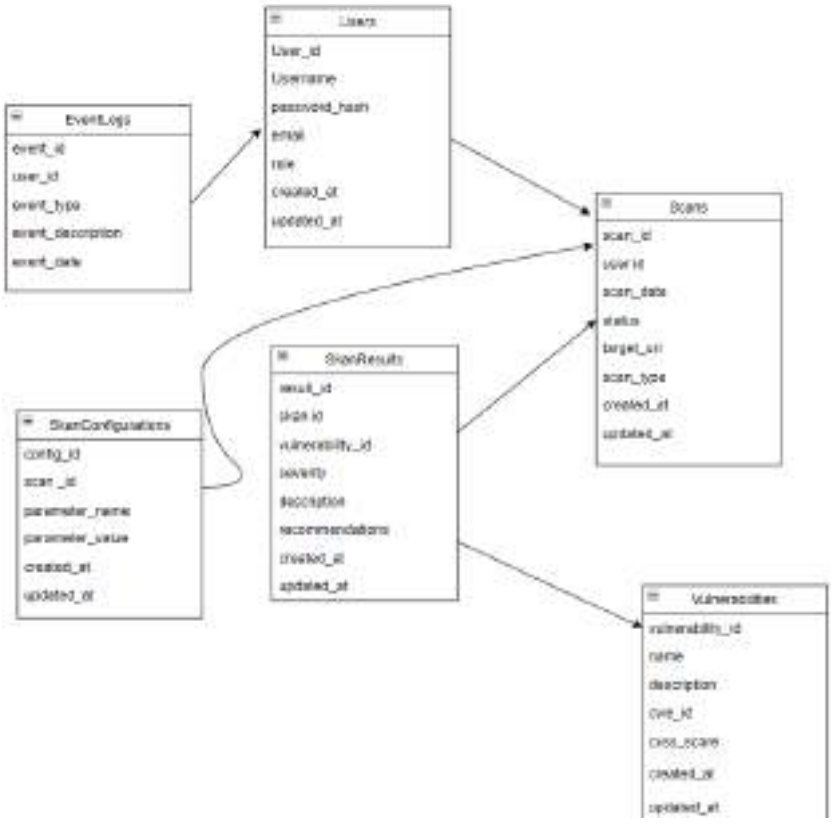


Рисунок 2.4 – Діаграма логічної схеми бази даних

Діаграма відображає логічні зв’язки між таблицями.

2.3 Аналіз та вибір технологій і методів реалізації WAF

Вибираючи технології та методи для система виявлення вразливостей вебзастосунків, необхідно враховувати кілька факторів. Інтеграція автоматизованого тестування на проникнення з використанням різних сканерів уразливостей, таких як Arachni та OWASP ZAP, може підвищити точність виявлення шляхом об’єднання їхніх результатів у консолідований звіт про вразливості. Цей підхід використовує алгоритми автоматизації та комбіновані методи для зменшення помилкових спрацьовувань і підвищення ефективності виявлення, що робить його вкрай важливим для надійної безпеки вебдодатків (MDPI).

Для розробки бази даних я використувавала MS SQL [4]. Основні переваги цієї програми полягають в тому, що SQL Server відомий своєю високою продуктивністю, особливо при роботі із великими обсягами даних, даний сервер надає ряд заходів для забезпечення надійності та безпеки даних, таких як резервне копіювання, реплікація, аунтефікація та авторизація, а також не меньш важливо, що даних сервер надає механіжм шифрування даних.

SQL [16] має глибоку інтеграцію з іншими продуктами та сервісами Microsoft, такими як Windows Server, .NET Framework, Azure, Excel та інші. Це дозволяє легко використовувати його в екосистемі Microsoft та інтегрувати з іншими рішеннями. Цей сервер підтримує різноманітні типи даних, включаючи стандартні типи даних SQL, такі як рядки, числа та дати, а також спеціалізовані типи даних, такі як XML, JSON,[26] географічні дані тощо. Загалом, MS SQL Server є потужним і надійним рішенням для управління базами даних, яке відповідає вимогам мого майбутього ПЗ.

Для реалізації функціональної частини, тобто написання основного коду програмного забезпечення мій вибір був між такими мовами програмування:

C++, Java, C# та Python, але серед них всіх найзручнішою та більш зрозумілішою для мене є C#. C# — це мова програмування, популярність якої зростає з кожним роком. Вона відзначається спрощенням коду завдяки

перевантаженню функцій і операторів, високою безпекою, можливістю використання об'єктно-орієнтованого підходу та управлінням ресурсами за допомогою RAII[25]. Курси програмування на C# допоможуть швидко освоїти особливості мови та навчитися писати код правильно.

Розробники, які володіють цією мовою, можуть створювати продукти найвищої якості. C# підходить для розробки як класичних, так і вебдодатків. Уміння створювати додатки є важливою перевагою в резюме девелопера, оскільки веб-розробка сьогодні дуже затребувана. Знання C# дозволяє стати затребуваним фахівцем із високим рівнем зарплати, забезпечуючи собі гарне майбутнє. Серед багатьох мов програмування, C# користується великою популярністю як серед початківців, так і серед досвідчених фахівців. Успіх цієї мови пояснюється тим, що програмістам легко виражати свої ідеї. C# здобув популярність завдяки зручній делегації, атрибутам та анонімним функціям. C# також підтримує перевантаження операторів і статичну типізацію. Простота полягає у доступності основного синтаксису. Якщо ви вивчали інформатику у школі, то C# буде зрозумілим для вас. Важливо мати базове розуміння англійської мови. Без цього програмування може здатися складним. Тому рекомендується також записатися на курси з англійської мови. .NET надає широкий спектр функціональності, який доповнюється синтаксичними конструкціями. З одного боку, .NET може здаватися простим, але при уважному розгляді стає очевидним, що він підходить для розробки складних систем. Важливо зазначити, що існує велика бібліотека готових рішень у формі nuget-пакетів, які можуть бути використані для різних потреб. Корпорація Microsoft відіграє ключову роль у розвитку мови. Програмісти можуть бути впевнені в постійному вдосконаленні та оновленнях. Нові технології та оновлені версії випускаються регулярно, що спрощує та зручніше робить життя розробників у вирішенні складних завдань. Пункт, який високо цінується програмістами. Завдяки C Sharp можна успішно розробити як стандартні програми для вебсайтів, так і сучасні мобільні додатки. Завдяки наявності величезного розмаїття інструментів і бібліотек, розробити можна навіть нейромережі.

Я для себе визначила кілька переваг цієї мови програмування над іншими, для прикладу C# має більш простий синтаксис порівняно з C++, що робить його більш доступним для початківців та зменшує ймовірність помилок. Він використовує автоматичне управління пам'яттю через збирач мусору, що спрощує роботу з динамічною пам'яттю. Також C# може бути використаний для розробки крос-платформенних додатків завдяки платформі .NET Core/.NET 5 та Xamarin. Це дозволяє створювати додатки, які можуть працювати на різних операційних системах, включаючи Windows, macOS та Linux. НЕ менши важливим є те, що C# пропонує вищий рівень абстракції порівняно з C++, що дозволяє розробникам швидше створювати додатки та уникати низькорівневих проблем, таких як пам'ять, показники тощо.

Отже, хоча C# і C++[24] мають свої власні сфери застосування,особисто для себе та для реалізації свого прозрамного забезпечення я обрала саме C#.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ

3.1 Розробка бази даних

Для створення цього програмного забезпечення була вибрана система управління базами даних Microsoft SQL Server. Даний сервер має свій зрозумілий інтерфейс та основні операції без яких не буде можливо зреалізувати та спроектувати жодну БД. До основних операцій належать SELECT, INSERT, UPDATE, DELETE [19].

Операція SELECT вибирає дані з бази даних відповідно до заданих умов і параметрів і дозволяє отримати ці дані для подальшого використання або відображення.

Принцип роботи операції INSERT полягає у тому, що вона дозволяє користувачеві додавати нові записи до вказаної таблиці бази даних, вказуючи значення для кожного стовпця, який потрібно заповнити. Після цього введення дані зберігаються у вказаній таблиці.

Робота UPDATE [23]полягає у тому, що вона здійснює зміни в існуючих даних в таблиці бази даних. Користувач вказує таблицю, яку слід оновити, а також нові значення для одного або кількох стовпців. Після виконання операції UPDATE вказані дані відповідно змінюються у таблиці.

Операції DELETE видаляє дані з таблиці бази даних. Користувач вказує таблицю, з якої потрібно видалити записи, а також можливі умови, які обмежують обсяг видаляємої інформації. Після виконання операції DELETE відповідні записи вилучаються з таблиці.

Інтерфейс програми Microsoft SQL Server може залежати від конкретної версії продукту та використовуваних інструментів, але незважаючи на це, завжди є очновні компоненти, які є незмінні, до них відноситься графічний інтерфейс користувача (GUI) тобто SQL Server Management Studio (SSMS), він надає широкі можливості ля адміністрування та розробки баз даних Microsoft SQL Server. У SSMS ви можете виконувати SQL-запити, адмініструвати сервери та бази даних, переглядати та редагувати дані, налаштовувати безпеку та багато іншого.

Налаштовувати різні параметри та параметри конфігурації для SQL Server, такі як мережеві з'єднання, служби та додаткові налаштування можна за допомогою SQL Server Configuration Manager. Засоби для створення та управління базами даних розробки і тестування схем баз даних, написання збережених процедур, керування версіями та інше надає середовище розробки SQL Server Data Tools (SSDT). Для моніторингу та аналізу діяльності сервера баз даних, включаючи виконані SQL-запити, події та звернення до ресурсів сервера використовується SQL Server Profiler. Для створення, управління та розповсюдження різноманітних звітів із використанням даних з SQL Server, використовується SQL Server Reporting Services (SSRS).

Після вибору середовища розгортання та запуску системи керування базами даних, важливо обрати спосіб створення та наповнення бази даних.

Для взаємодії серверної частини додатку з базою даних було вибрано технологію Entity Framework Core. Вона надає засоби для визначення різних способів підключення до бази даних, опису сутностей та їхньої взаємодії.

Розробники Entity Framework Core[20] використовують підхід "code-first" до роботи з базами даних. Цей підхід передбачає опис усіх сутностей бази даних у вигляді класів мови програмування C#. Після цього виконується налаштування зв’язків між цими сутностями та реєстрація усіх класів.

Таким чином, проведено аналіз спроектованої бази даних та описано всі необхідні класи мови програмування C# для збереження відповідних даних та їх взаємодії. Далі розглянемо ці класи докладніше.

Клас User визначає сутність користувача та зберігає в ньому основну інформацію. Ця інформація включає унікальний ідентифікатор користувача в таблиці бази даних, ім'я, електронну пошту, роль користувача, хеш паролю, рейтинг користувача, посилання на фото профілю та статус акаунта.

Код програми в якому реалізовано клас User:

```
namespace YourNamespace
{
    public class User
    {
```

```
        [Key]
        [DatabaseGenerated(DatabaseGeneratedOption.Identity)]
        public int UserId { get; set; } // Унікальний ідентифікатор користувача
(PK)

        [Required]
        [MaxLength(50)]
        public string Username { get; set; } // Ім'я користувача

        [Required]
        public string PasswordHash { get; set; } // Хеш пароля

        [Required]
        [EmailAddress]
        public string Email { get; set; } // Електронна адреса користувача

        [Required]
        public string Role { get; set; } // Роль користувача в системі

        [Required]
        public DateTime CreatedAt { get; set; } // Дата та час створення
облікового запису користувача

        [Required]
              public DateTime UpdatedAt { get; set; } // Дата та час останнього
оновлення даних користувача
          }
      }
```

Цей код визначає клас User, що відповідає сутності користувача в базі даних. Поля класу представляють атрибути користувача, такі як ідентифікатор, ім'я користувача, хеш пароля, електронна адреса, роль, дата створення та останнього оновлення облікового запису. Клас використовує атрибути з простору імен SystemComponentModelDataAnnotations для вказівки характеристик полів бази даних, таких як первинний ключ, обов'язковість та максимальна довжина.

Наступним класом, реалізованим у програмі є клас Scans у якому зберігається та описується інформацію по кожному скануванню програми.

Код реалізації цього класу виглядає, як наведено нижче:

```
public class Scan
    {
        [Key]
        [DatabaseGenerated(DatabaseGeneratedOption.Identity)]
        public int ScanId { get; set; } // Унікальний ідентифікатор сканування
(PK)

        [ForeignKey("User")]
        public int UserId { get; set; } // Ідентифікатор користувача (FK)
```

```
        [Required]
        public DateTime ScanDate { get; set; } // Дата та час початку сканування

        [Required]
        [MaxLength(50)]
        public string Status { get; set; } // Статус сканування

        [Required]
        [Url]
        public string TargetUrl { get; set; } // Цільовий URL для сканування

        [Required]
        [MaxLength(50)]
        public string ScanType { get; set; } // Тип сканування

        [Required]
        public DateTime CreatedAt { get; set; } // Дата створення запису про
сканування

        [Required]
        public DateTime UpdatedAt { get; set; } // Дата останнього оновлення
даних

        // Навігаційна властивість для зв'язку з таблицею Users
        public virtual User User { get; set; }
          }
      }
```

Даний клас Scan представляє сутність сканування в базі даних. Поля класу відповідають атрибутам сканування, такими як унікальний ідентифікатор, ідентифікатор користувача, дата і час сканування, статус, цільовий URL, тип сканування, дата створення і дата останнього оновлення. Клас також включає навігаційну властивість User[21], що дозволяє встановити зв'язок з таблицею Users через зовнішній ключ UserId.

Для того щоб результати по кожному скануванню зберігалися в базі даних реалізовано клас ScanResult.

Код реалізації:

```
public class ScanResult
    {
        [Key]
        [DatabaseGenerated(DatabaseGeneratedOption.Identity)]
        public int ResultId { get; set; } // Унікальний ідентифікатор результату
сканування (PK)

        [ForeignKey("Scan")]
        public int ScanId { get; set; } // Ідентифікатор сканування (FK)

        [ForeignKey("Vulnerability")]
```

```
        public int VulnerabilityId { get; set; } // Ідентифікатор виявленої
вразливості (FK)

        [Required]
        [MaxLength(50)]
        public string Severity { get; set; } // Рівень критичності вразливості

        [Required]
        public string Description { get; set; } // Опис вразливості

        public string Recommendations { get; set; } // Рекомендації щодо усунення
вразливості

        [Required]
        public DateTime CreatedAt { get; set; } // Дата створення запису про
результат сканування

        [Required]
        public DateTime UpdatedAt { get; set; } // Дата останнього оновлення
даних

        // Навігаційні властивості для зв'язку з таблицями Scans та
Vulnerabilities
        public virtual Scan Scan { get; set; }
        public virtual Vulnerability Vulnerability { get; set; }
          }
```

Даний клас містить поля, які відповідають атрибутам результату сканування, такі як унікальний ідентифікатор, ідентифікатори сканування та вразливості, рівень критичності, опис, рекомендації, дати створення та оновлення. Клас також включає навігаційні властивості для встановлення зв'язків з таблицями Scans та Vulnerabilities.

У вищеописаному класі ScanResult[17] було згадано наступний не менш важливий клас Vulnerabilities за допомогою якого у програмі зберігається інформацію про вразливості опрацьованих вебдодатків.

Код реалізації класу:

```
public class Vulnerability
    {
        [Key]
        [DatabaseGenerated(DatabaseGeneratedOption.Identity)]
        public int VulnerabilityId { get; set; } // Унікальний ідентифікатор
вразливості (PK)

        [Required]
        [MaxLength(100)]
        public string Name { get; set; } // Назва вразливості

        [Required]
        public string Description { get; set; } // Опис вразливості
```

```
        [Required]
        [MaxLength(50)]
        public string CweId { get; set; } // Ідентифікатор загальної слабкості
(CWE)

        [Required]
        [Range(0, 10)]
        public double CvssScore { get; set; } // Оцінка CVSS

        [Required]
        public DateTime CreatedAt { get; set; } // Дата створення запису про
вразливість

        [Required]
        public DateTime UpdatedAt { get; set; } // Дата останнього оновлення
даних
    }
}
```

Поля класу відповідають атрибутам вразливості, такими як унікальний ідентифікатор, назва, опис, ідентифікатор CWE[16], оцінка CVSS, дата створення та дата останнього оновлення. Клас використовує атрибути з простору імен System.ComponentModel.DataAnnotations та DataAnnotations.Schema для вказівки характеристик полів бази даних.

Наступним у зв’язку бази даних є клас ScanConfiguration. Призначення цього класу полягає в тому, що у ньому представлено конфігураційні параметри для сканувань у базі даних.

Код реалізації класу:

```
public class ScanConfiguration
    {
        [Key]
        [DatabaseGenerated(DatabaseGeneratedOption.Identity)]
        public int ConfigId { get; set; } // Унікальний ідентифікатор
конфігурації (PK)

        [ForeignKey("Scan")]
        public int ScanId { get; set; } // Ідентифікатор сканування (FK)

        [Required]
        [MaxLength(100)]
        public string ParameterName { get; set; } // Назва параметра конфігурації

        [Required]
        public string ParameterValue { get; set; } // Значення параметра
конфігурації

        [Required]
        public DateTime CreatedAt { get; set; } // Дата створення запису про
конфігурацію

        [Required]
```

```
        public DateTime UpdatedAt { get; set; } // Дата останнього оновлення
даних

        // Навігаційна властивість для зв'язку з таблицею Scans
        public virtual Scan Scan { get; set; }
    }
}
```

Поля класу відповідають атрибутам конфігурації, такими як унікальний ідентифікатор, ідентифікатор сканування, назва параметра, значення параметра, дата створення та дата останнього оновлення. Клас[15] також включає навігаційну властивість Scan, що дозволяє встановити зв'язок з таблицею Scans через зовнішній ключ ScanId.

Останнім а заключним класом є EventLog, клас подій в якому записується дані по кожній події, яка опрацьовується програмою.

Код класу:

```
public class EventLog
    {
        [Key]
        [DatabaseGenerated(DatabaseGeneratedOption.Identity)]
        public int EventId { get; set; } // Унікальний ідентифікатор події (PK)

        [ForeignKey("User")]
        public int UserId { get; set; } // Ідентифікатор користувача (FK)

        [Required]
        [MaxLength(50)]
        public string EventType { get; set; } // Тип події

        [Required]
        public string EventDescription { get; set; } // Опис події

        [Required]
        public DateTime EventDate { get; set; } // Дата та час події

        // Навігаційна властивість для зв'язку з таблицею Users
        public virtual User User { get; set; }
    }
}
```

3.2 Розробка програмних модулів

Основою системи виявлення вразливості вебдодатків є функціонал самої системи до якого відноситься аудит безпеки, перехоплення та аналіз трафіку,

сканування на вразливість, автоматизація тестування, інтеграція з іншими системами, керування користувачами, налаштування системи. Аудит безпеки – це критичний процес, якиє є спрямований на оцінку безпеки самої системи, який поділяється на кілька етапів, кожен з яких має на меті виявлення потенційних ризиків та надання рекомендацій щодо їх усунення. До основних етапів можна віднести: планування, збір та аналіз інформації, тестування технічних засобів, оцінка процесів безпеки. Призначення аудиту безпеки полягає у захисті, підвищенні рівня безпеки, заощадження ресурсів та відповідність стандартам безпеки. Для самої реалізації аудиту безпеки у програмі було використано кілька методів, код яких буде наведено нижче.

Першим методом є метод перевірки доступності вебдодатків у системі. Використовується для визначення, чи працює вебдодаток в системі та чи є він доступним для користувачів. Він дозволяє моніторити доступність вебдодатків та виявлення можливих проблем з їх роботою та оперативного реагування на них.

Код реалізації цього класу:

```
public List<WebApp> CheckWebAppsAvailability()
{
    List<WebApp> availableWebApps = new List<WebApp>();

    // Логіка для перевірки доступності вебдодатків
    // Додавання доступних вебдодатків до списку availableWebApps

    return availableWebApps;
}
```

Для того щоб не виникало проблем із актуальністю версії вебдодатка та його компонентів використовую наступний метод перевірки:

```
public List<WebApp> CheckWebAppVersions(List<WebApp> webApps)
{
    List<WebApp> outdatedWebApps = new List<WebApp>();

    // Логіка для перевірки актуальності версій вебдодатків та їхніх компонентів
    // Додавання застарілих вебдодатків до списку outdatedWebApps

    return outdatedWebApps;
}
```

Наступний метод слугує для аналізу журналів подій та виявлення підозрілих або несподіваних дій

```
public List<EventLog> AnalyzeEventLogs()
{
    List<EventLog> suspiciousEvents = new List<EventLog>();

    // Логіка для аналізу журналів подій та виявлення підозрілих або несподіваних
дій
    // Додавання підозрілих подій до списку suspiciousEvents

    return suspiciousEvents;
}
```

Всі методи методи можна розширити та налаштувати відповідно до конткретних вимог даної системи виявлення вразливості вебдодатків.

Для перехоплення та аналізу трафіку у системі виявлення вразливостей вебдодатків використовую бібліотеку SharpPcap тому, що вона надає можливості для перехоплення та обробки мережевих пакетів у середовищі .NET. [11]

SharpPcap [5] дозволяє прослуховувати мережевий трафік, що проходить через вибраний мережевий інтерфейс. Це дозволяє аналізувати мережеві пакети у реальному часі. Також підтримує різні типи мережевих пакетів, такі як Ethernet, IP, TCP, UDP що дозволяє аналізувати різноманітні мережеві протоколи. НЕ менш важливим аспектом, чому було вибрано саме цю бібліотеку, це те що harpPcap має простий і легко використовуваний API, який дозволяє розробникам швидко розпочати роботу з перехопленням мережевого трафіку у своїх програмах.Для узагальнення можна сказати, що SharpPcap є потужним інструментом для аналізу мережевого трафіку у середовищі .NET,[12] який дозволяє створювати різноманітні програми для моніторингу та аналізу мережі. Тому нижче наведений код реалізовано для перехоплення пакету даних із мережевого інтерфейсу та їхнього аналізу:

```
class Program
{
    static void Main(string[] args)
    {
        // Вибір мережевого інтерфейсу для прослуховування
        var device = CaptureDeviceList.Instance.FirstOrDefault();
```

```
        if (device == null)
        {
            Console.WriteLine("Не знайдено жодного мережевого інтерфейсу.");
            return;
        }

        // Встановлення фільтра для перехоплення HTTP-трафіку (приклад)
        device.Filter = "tcp port 80";

        // Початок прослуховування мережі
        device.OnPacketArrival += (sender, e) =>
        {
            var packet = Packet.ParsePacket(e.Packet.LinkLayerType,
e.Packet.Data);
            var tcpPacket = (TcpPacket)packet.Extract(typeof(TcpPacket));

            if (tcpPacket != null && tcpPacket.PayloadData.Length > 0)
            {
                Console.WriteLine("Перехоплено пакет:");
                Console.WriteLine(tcpPacket.PrintHex());
            }
        };

        device.Open(DeviceMode.Promiscuous);
        device.StartCapture();

        Console.WriteLine("Прослуховування почалося. Натисніть будь-яку клавішу
для завершення...");
        Console.ReadKey();

        // Зупинка прослуховування мережі та звільнення ресурсів
        device.StopCapture();
        device.Close();
    }
}
```

Наступним кроком є реалізація сканування та виявлення самих вразливостей вед-додатка, переде самим скануванням, потрібно обрати тип сканування у програмі це виглядає як показано на рисунку 3.1.

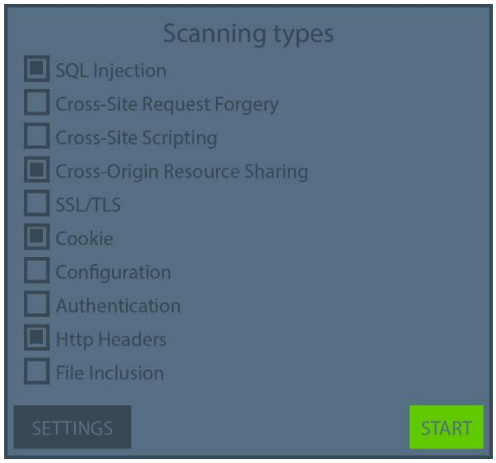


Рисунок 3.1 – Типи сканування

Після вибору типу сканування, сам процес у програмі виглядає так як це показано на рисунку 3.2.

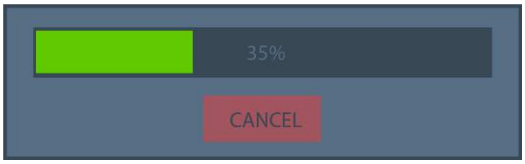


Рисунок 3.2 – Процес сканування

Код реалізації даного класу подано нижче:

```
class Program
{
    static void Main(string[] args)
    {
        // Виберіть мережевий інтерфейс для перехоплення пакетів
        var device = CaptureDeviceList.Instance[0]; // Виберіть перший доступний
мережевий інтерфейс

        // Встановіть фільтр для перехоплення HTTP трафіку
        device.Filter = "tcp port 80";

        // Встановіть обробник пакетів для обробки кожного перехопленого пакета
        device.OnPacketArrival += (sender, e) =>
        {
            var packet = Packet.ParsePacket(e.Packet.LinkLayerType,
e.Packet.Data);
            var tcpPacket = (TcpPacket)packet.Extract(typeof(TcpPacket));

            // Перевірте, чи пакет містить HTTP дані
            if (tcpPacket != null && tcpPacket.PayloadData.Length > 0)
            {
                var httpData =
System.Text.Encoding.ASCII.GetString(tcpPacket.PayloadData);

                // Перевірте, чи HTTP дані містять потенційно вразливі запити або
відповіді
                if (httpData.Contains("SQL Injection"))
                {
                    Console.WriteLine("Знайдено потенційну вразливість SQL
Injection");
                    Console.WriteLine(httpData);
                }
                else if (httpData.Contains("XSS Attack"))
                {
                    Console.WriteLine("Знайдено потенційну вразливість XSS
Attack");
                    Console.WriteLine(httpData);
                }
            }
        };

        // Відкрити пристрій для перехоплення трафіку
        device.Open(DeviceMode.Promiscuous);
```

```
        // Почати захоплення пакетів
        device.StartCapture();

        Console.WriteLine("Натисніть будь-яку клавішу для завершення...");
        Console.ReadKey();

        // Зупинити захоплення пакетів та закрити пристрій
        device.StopCapture();
        device.Close();
    }
}
```

Дана частина програми дозволяє перехоплювати HTTP трафік на вказаному мережевому інтерфейсі, а потім аналізувати цей трафік на наявність потенційних вразливостей, таких як SQL Injection або XSS Attack.

Сканування системи виявлення вразливості веб- додатків полягає в ідентифікації потенційних слабкостей, уразливостей та проблем безпеки у вебдодатках . Цей процес є ключовим етапом в процесі забезпечення безпеки вебдодатків та допомагає уникнути серйозних проблем безпеки та порушень.

Реалізацію у програмі, наведено нижче:

```
class Program
{
    static void Main(string[] args)
    {
        // Виконати сканування системи виявлення вразливості вебдодатків
        ScanVulnerabilitySystem();

        Console.WriteLine("Сканування завершено.");
    }

    static void ScanVulnerabilitySystem()
    {
        // Виконати підготовчі дії перед скануванням (наприклад, налаштування
мережевого інтерфейсу)

        // Запустити сканер вразливостей
        Console.WriteLine("Запуск сканера вразливостей...");
        RunVulnerabilityScanner();

        // Очікувати завершення сканування протягом певного часу
        int timeoutSeconds = 300; // Наприклад, 5 хвилин
        Console.WriteLine($"Очікування завершення сканування протягом
{timeoutSeconds} секунд...");
        Thread.Sleep(timeoutSeconds * 1000); // Перетворення секунд на
мілісекунди

        // Обробити результати сканування і прийняти відповідні дії
        ProcessScanResults();
    }
```

```
    static void RunVulnerabilityScanner()
    {
        // Код для запуску сканера вразливостей
        Console.WriteLine("Сканер вразливостей запущено.");
    }

    static void ProcessScanResults()
    {
        // Код для обробки результатів сканування
        Console.WriteLine("Результати сканування оброблені.");
    }
}
```

Інтеграція системи полягає у забезпеченнні взаємодії цієї системи із компонентами або сервісами, які допомагають виявляти, аналізувати та реагувати на потенційні загрози безпеки.[9] Також потрібно розуміти, що інтеграція із іншими системами може включати передачу даних про виявлені вразливості або події безпеки, автоматизацію реагування на ці події, а також обмін інформацією з іншими інструментами безпеки. Ці всі функції реалізовано у програмі, код наведено нижче:

```
class VulnerabilityIntegration
{
    private readonly HttpClient _client;

    public VulnerabilityIntegration()
    {
        _client = new HttpClient();
    }

    // Метод для передачі даних про виявлені вразливості іншим системам
    public async Task SendVulnerabilityData(string vulnerabilityData)
    {
        try
        {
            var response = await
_client.PostAsync("https://example.com/vulnerability", new
StringContent(vulnerabilityData));
            response.EnsureSuccessStatusCode();
            Console.WriteLine("Дані про вразливість успішно передані.");
        }
        catch (HttpRequestException e)
        {
            Console.WriteLine($"Помилка під час передачі даних про вразливість:
{e.Message}");
        }
    }

    // Метод для автоматизованого реагування на виявлені вразливості
    public async Task AutomateResponseToVulnerability(string vulnerabilityId)
    {
        // Логіка автоматичного реагування на виявлену вразливість
```

```
        Console.WriteLine($"Запускається автоматична реакція на вразливість з ID:
{vulnerabilityId}");
        await Task.Delay(2000); // Приклад часової затримки для імітації процесу
        Console.WriteLine("Автоматична реакція завершена.");
    }

    // Метод для обміну інформацією з іншими системами безпеки
    public async Task ExchangeSecurityInformation(string securityInfo)
    {
        try
        {
            var response = await
_client.PostAsync("https://example.com/security/info", new
StringContent(securityInfo));
            response.EnsureSuccessStatusCode();
            Console.WriteLine("Інформація про безпеку успішно передана.");
        }
        catch (HttpRequestException e)
        {
            Console.WriteLine($"Помилка під час передачі інформації про безпеку:
{e.Message}");
        }
    }
}

class Program
{
    static async Task Main(string[] args)
    {
        VulnerabilityIntegration integration = new VulnerabilityIntegration();

        // Приклад виклику методів для інтеграції з іншими системами
        await integration.SendVulnerabilityData("Дані про виявлену вразливість");
        await integration.AutomateResponseToVulnerability("123456");
        await integration.ExchangeSecurityInformation("Інформація про безпеку");
    }
}
```

Створення користувачів можливе лише за допомогою адміністратора програми, лише він має можливість створювати нові облікові записи та надавати їм доступ до системи виявлення вразливості. На рисунку 3.3 зображено вхід у систему.

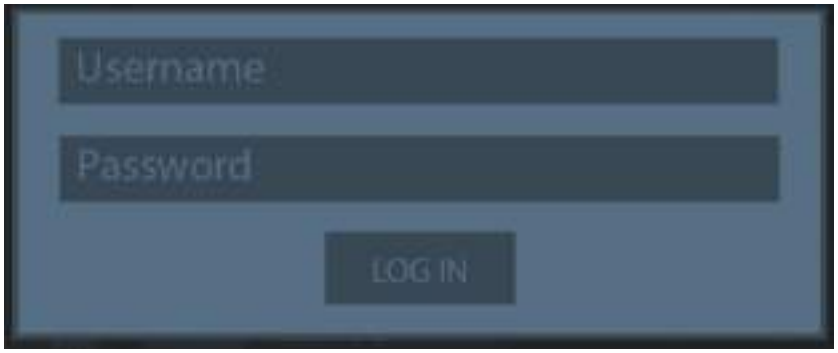


Рисунок 3.3 – Авторизація користувача у систему

Також користувачам можуть бути надані різні рівні доступу до функціональності системи відповідно до їхніх ролей та повноважень.[10] Система може вести журнал дій користувачів ждля відстеження їхньої активності в системі, що дозволяє виявляти ненормальну або підозрілу поведінку. Адміністратор системи може мати можливість скидати паролі користувачів, видаляти або заблокувати облікові записи у випадку потреби або забезпечення безпеки. Система виявлення вразливостей може інтегруватися з системами ідентифікації та автентифікації, такими Active Directory або LDAP, для управління користувачами та їх обліковими записами.

Код реалізації:

```
public class User
{
    public int UserId { get; set; }
    public string Username { get; set; }
    public string PasswordHash { get; set; }
    public string Email { get; set; }
    public string Role { get; set; }
    public DateTime CreatedAt { get; set; }
    public DateTime UpdatedAt { get; set; }
}

public class UserManager
{
    public void CreateUser(string username, string passwordHash, string email,
string role)
    {
        // Логіка для створення нового користувача
        var newUser = new User
        {
            UserId = GenerateUserId(), // Генеруємо унікальний ідентифікатор
користувача
            Username = username,
            PasswordHash = passwordHash,
            Email = email,
            Role = role,
            CreatedAt = DateTime.Now,
            UpdatedAt = DateTime.Now
        };

        Console.WriteLine($"Користувач {username} успішно створений.");
    }

    public void UpdateUser(int userId, string username, string passwordHash,
string email, string role)
    {
        // Логіка для оновлення інформації про користувача
        var user = GetUserById(userId);
        if (user != null)
        {
            user.Username = username;
```

```
            user.PasswordHash = passwordHash;
            user.Email = email;
            user.Role = role;
            user.UpdatedAt = DateTime.Now;
            Console.WriteLine($"Інформація про користувача з ID {userId} успішно
оновлена.");
        }
        else
        {
            Console.WriteLine($"Користувача з ID {userId} не знайдено.");
        }
    }

    public void DeleteUser(int userId)
    {
        // Логіка для видалення користувача
        var user = GetUserById(userId);
        if (user != null)
        {
            // Виконати видалення користувача з бази даних або іншого сховища
            Console.WriteLine($"Користувач з ID {userId} успішно видалений.");
        }
        else
        {
            Console.WriteLine($"Користувача з ID {userId} не знайдено.");
        }
    }

    public User GetUserById(int userId)
    {
        // Логіка для отримання користувача за його ідентифікатором з бази даних
або іншого сховища
        // Повертаємо знайденого користувача або null, якщо користувач не
знайдений
        return null;
    }

    private int GenerateUserId()
    {
        // Логіка для генерації унікального ідентифікатора користувача
        // Зазвичай використовується для генерації нового значення ID при
створенні нового користувача
        return new Random().Next(1000, 9999); // Приклад генерації випадкового
числа у діапазоні
    }
}
```

Розуміння того, які саме вразливості потрібно виявляти, і які додатки або сервіси будуть піддаватися скануванню допоможе визначити ціль та обсяг системи. Вибір потрібних інструментів для сканування і виявлення вразливостей. Це може бути комбінація автоматизованих[8] сканерів вразливостей, ручного аудиту коду, перехоплення трафіку і аналізу даних тощо. Налаштування параметрів сканування, таких як глибина сканування, частота сканування, перелік вразливостей для перевірки тощо. Важливо враховувати можливості і обмеження

обраного сканера. Постійне оновлення і вдосконалення налаштувань системи виявлення вразливостей на основі нових загроз і вразливостей, а також отриманих результатів сканування і аудиту.[7] Проведення тестів і аудиту налаштувань системи виявлення вразливостей для визначення їхньої ефективності і відповідності вимогам безпеки. Налаштування системи виявлення вразливостей вебдодатків є процесом, що вимагає уваги до деталей і постійного вдосконалення, щоб забезпечити ефективність захисту системи від потенційних загроз та нападів на вебдодатки.

Код, що буде наведено нижче, демонструє налаштування системи виявлення вразливості:

```
namespace VulnerabilityDetectionSystem
{
    public class VulnerabilityDetectionConfiguration
    {
        public string ScannerType { get; set; }
        public int ScanDepth { get; set; }
        public bool IncludeThirdPartyLibraries { get; set; }

        public VulnerabilityDetectionConfiguration(string scannerType, int
scanDepth, bool includeThirdPartyLibraries)
        {
            ScannerType = scannerType;
            ScanDepth = scanDepth;
            IncludeThirdPartyLibraries = includeThirdPartyLibraries;
        }

        public void DisplayConfiguration()
        {
            Console.WriteLine("Vulnerability Detection Configuration:");
            Console.WriteLine($"Scanner Type: {ScannerType}");
            Console.WriteLine($"Scan Depth: {ScanDepth}");
            Console.WriteLine($"Include Third-Party Libraries:
{(IncludeThirdPartyLibraries ? "Yes" : "No")}");
        }
    }

    class Program
    {
        static void Main(string[] args)
        {
            // Creating a new instance of VulnerabilityDetectionConfiguration
            VulnerabilityDetectionConfiguration config = new
VulnerabilityDetectionConfiguration("OWASP ZAP", 3, true);

            // Displaying the initial configuration
            Console.WriteLine("Initial Configuration:");
            config.DisplayConfiguration();

            // Modifying the configuration
            config.ScanDepth = 5;
```

```
            config.IncludeThirdPartyLibraries = false;

            // Displaying the modified configuration
            Console.WriteLine("\nModified Configuration:");
            config.DisplayConfiguration();

            Console.ReadLine();
        }
    }
}
```

У даній частині коду було створено клас VulnerabilityDetectionConfiguration, який містить властивості для типу сканера, глибини сканування та прапорець для включення сторонніх бібліотек. У методі Main створено екземпляр цього класу, відображаємо початкову конфігурацію, змінюємо деякі параметри та відображаємо змінену конфігурацію.

3.3 Технічні характеристики

Технічні характеристики системи виявлення вразливостей вебдодатків зазвичай включають здатність до глибокого сканування коду на предмет уразливостей, таких як SQL ін'єкції, XSS, CSRF та інші. Система може використовувати різні методи аналізу, включаючи статичний та динамічний аналіз коду. Важливими параметрами[6] є швидкість сканування, точність виявлення вразливостей, можливість інтеграції з іншими інструментами розробки, а також підтримка різноманітних програмних мов і фреймворків. Ключовим елементом є також забезпечення надійних звітів та рекомендацій щодо усунення знайдених вразливостей. Важливою характеристикою є здатність системи працювати з великим обсягом додатків та швидко реагувати на зміни в навантаженні. Наявність зручного інтерфейсу для адміністраторів та користувачів системи для налаштування та візуалізації результатів сканування. Можливість реєструвати та аналізувати дії користувачів, події системи та результати сканування для подальшого аналізу та аудиту. Система сама повинна бути захищена від атак, оскільки сама може бути об'єктом атак. Система може бути

здатна виявляти вразливості в вебдодатках, написаних на різних мовах програмування (наприклад, Java, Python, PHP, C#) та побудованих на різних фреймворках (наприклад, Spring, Django, Laravel, ASP.NET). Вона може використовувати різні методи для виявлення вразливостей, включаючи аналіз вихідного коду, тестування зовнішнього вигляду, перехоплення та аналіз трафіку тощо.

3.4 Вибір та обґрунтування методів тестування ПЗ

При розробці систем виявлення вразливостей вебдодатків використовуються різні види тестувань.

Першим та найпоширенішим є Статичний аналіз коду (SAST) [6] - перевірка коду на вразливості без виконання програми. Під час статичного аналізу коду програма перевіряється без його активного виконання, шляхом аналізу джерел коду, конфігураційних файлів та інших ресурсів.

Основна ідея полягає у виявленні потенційних проблем у вихідному коді, які можуть призвести до вразливостей безпеки або інших проблем. Це можуть бути такі проблеми, як використання небезпечних функцій, неправильна обробка введених даних, можливість SQL-ін'єкцій, кросс-сайтового скриптування (XSS) та інші.

Після виявлення потенційних проблем програміст може виправити їх, зменшивши ризик появи вразливостей у програмному забезпеченні перед його виконанням та релізом.

Системи статичного аналізу коду можуть використовувати різні методи та алгоритми для виявлення проблем, від простих правил перевірки до складних аналітичних моделей. Цей процес може бути автоматизованим та інтегрованим у розробницьке середовище для забезпечення швидкості та ефективності перевірки коду.

Наступним видом тестування є динамічний аналіз коду (DAST) [6] тобто тестування вебдодатку в реальному часі на виявлення уразливостей. У процесі DAST система намагається здійснити реальні HTTP-запити та аналізує відповіді сервера на предмет виявлення потенційних вразливостей.

Основна ідея полягає в тому, щоб виконати тестування, симулюючи дії реального зловмисника, який намагається зламати або отримати несанкціонований доступ до вебдодатку. DAST може виявляти різноманітні види вразливостей, такі як кросс-сайтовий скриптінг (XSS), введення SQL-запитів, некоректна автентифікація та авторизація, використання слабких шифрувань, небезпечні налаштування сервера та інші.

Особливі переваги DAST включають можливість оцінки реального стану безпеки вебдодатку під час його роботи, ідентифікацію вразливостей, які можуть бути важко виявити іншими методами, і здатність виявляти проблеми, які виникають у результаті взаємодії різних компонентів системи.

Проте DAST також має свої обмеження. Наприклад, він може бути менш ефективним у виявленні проблем, які пов'язані зі слабким контролем доступу, недостатньою обробкою введених даних або іншими проблемами, які не виявляються під час активних HTTP-запитів.

Узагальнюючи, використання DAST разом з іншими методами, такими як статичний аналіз коду (SAST) та інші види тестування, може забезпечити більш повне тестування та покращити загальний рівень безпеки вебдодатків.

Інтерактивне тестування застосунків (IAST) полягає у комбінація статичного і динамічного аналізів, що виконується під час виконання коду. У відміну від інших методів тестування, які проводять аналіз коду перед або після виконання програми, IAST вбудовує агентів безпеки (security agents) безпосередньо в робоче середовище програмного забезпечення. Ці агенти моніторять виконання програми, аналізуючи вхідні та вихідні дані, а також взаємодію програми з системою. Інтерактивне тестування безпеки додатків (Interactive Application Security Testing, IAST) [25] — це термін, що охоплює інструменти, які поєднують переваги статичного тестування безпеки додатків

(SAST) і динамічного тестування безпеки додатків (DAST). Оскільки це загальний термін, інструменти IAST можуть значно відрізнятися у своїх підходах до тестування безпеки вебдодатків. Далі ми розглянемо, як ці інструменти з’явилися, яким чином вони виявляють уразливості, а також їх переваги та недоліки. Бізнеси, які добре розбираються у веб-безпеці, традиційно використовують ці два типи інструментів окремо. SAST використовується для перевірки коду компаніями, що розробляють власні вебдодатки, тоді як DAST частіше використовуються підприємствами, які мають веб-сторінки або вебдодатки.

Щоб полегшити роботу підприємств, виробники засобів безпеки вебдодатків зрозуміли, що можна об’єднати методи статичного і динамічного тестування для створення більш досконалих інструментів, які включають переваги обох підходів. Так з’явилося інтерактивне тестування безпеки додатків (IAST).

Основна проблема IAST полягає в тому, що ця ідея виникла у виробників інструментів SAST і DAST незалежно один від одного, що призвело до появи продуктів, які використовують один і той же термін, але значно різняться між собою. Наявні на ринку рішення IAST зазвичай не створені з нуля, а є розширенням традиційних сканерів вихідного коду або веб-сканерів вразливостей. Тому клієнти повинні бути впевнені, що обраний продукт відповідає їхнім потребам.

Основна перевага IAST [6] полягає в тому, що воно забезпечує більш повну та реалістичну оцінку безпеки застосунку, оскільки аналіз відбувається під час реального виконання програми в реальному середовищі. Це дозволяє виявляти вразливості, які можуть бути важко виявити за допомогою інших методів, таких як кросс-сайтовий скриптінг (XSS), SQL-ін'єкції та інші.

Проте, інтерактивне тестування застосунків також має свої обмеження, такі як високі вимоги до ресурсів системи, можливість впливу на продуктивність програми та обмеженість виявлення деяких типів вразливостей.

Узагальнюючи, IAST може бути ефективним методом тестування безпеки, який доповнює інші методи, такі як статичний аналіз коду (SAST) та динамічний

аналіз коду (DAST), забезпечуючи більш повне та реалістичне тестування безпеки програмного забезпечення.

Менш відомим є тестування залежностей (SCA). Це метод аналізу використовуваних бібліотек та фреймворків у програмному забезпеченні з метою виявлення потенційних вразливостей та проблем безпеки.

Основна ідея SCA полягає в тому, щоб ідентифікувати та оцінити використані бібліотеки та їх версії у вихідному коді або скомпільованому програмному забезпеченні. Після цього проводиться перевірка цих бібліотек на предмет відомих вразливостей та проблем безпеки, які можуть бути використані зловмисниками для атак.

Тестування залежностей може включати в себе такі дії:

- аналіз використаних бібліотек пошук та ідентифікація всіх використаних бібліотек та фреймворків у програмному забезпеченні;
- оцінка вразливостей перевірка кожної бібліотеки на предмет відомих вразливостей та проблем безпеки, які можуть вплинути на безпеку системи;
- виявлення вразливостей ідентифікація конкретних вразливостей у використовуваних бібліотеках та оцінка їх серйозності;
- надання рекомендацій надання рекомендацій щодо усунення виявлених вразливостей, таких як оновлення бібліотеки до безпечної версії або застосування патчів та виправлень.

Тестування залежностей доповнює інші методи тестування безпеки, такі як статичний аналіз коду (SAST)[4] та динамічний аналіз коду (DAST), забезпечуючи більш повне та комплексне тестування безпеки програмного забезпечення.

Поширеним методом сканування у кібезбезпеці вважається Фаззінг. Фаззінг — це методика тестування, яка використовується для виявлення вразливостей у програмному забезпеченні, особливо тих, які можуть бути використані для атак. Основна ідея полягає в тому, щоб ввести в програму непередбачувані або некоректні дані (фазз-вектори) та спостерігати, як вона реагує на ці дані. Основні етапи фаззінгу включають наступне:

- генерація фазз-векторів створення набору непередбачуваних або некоректних вхідних даних, які можуть бути надіслані до програми;

- введення фазз-векторів в програму фаззер відправляє фазз-вектори в програму для обробки;

- спостереження за реакцією програми фаззер відслідковує реакцію програми на кожен фазз-вектор, виявляючи можливі збої, виключення або інші аномалії;

- відлагодження та аналіз результатів якщо фаззер виявляє збій програми, розробники можуть використовувати цю інформацію для виправлення вразливості.[5]

Фаззінг може бути використаний для тестування різноманітних типів програмного забезпечення, включаючи вебдодатки, мобільні додатки, операційні системи, драйвери пристроїв та інше. Він є важливим інструментом у кібербезпеці для виявлення та усунення вразливостей до їх експлуатації зловмисниками.

3.5 Застосування обраного методу тестування на WAF

Для тестування системи виявлення вразливості вебдодатка, яка є описана у даній кваліфікаційної роботи, використано метод статистичного аналізу коду. У WAF статичний аналіз коду використовується для оцінки безпеки додатка на рівні джерела, тобто шляхом аналізу самого вихідного коду. Статичний аналіз дозволяє виявляти потенційні вразливості в коді, такі як XSS[3] (міжсайтовий скриптінг), SQL-ін'єкції, вразливості в обробці введених даних та інші. Аналізуючи виклики функцій та методів, можна виявити код, який може бути використаний для атак або зловмисницьких дій. Статичний аналіз допомагає виявити можливі проблеми з обробкою та збереженням конфіденційних даних, таких як паролі або особиста інформація користувачів. При розробці системи виявлення вразливостей вебдодатків, статичний аналіз коду став важливим інструментом для забезпечення безпеки та захисту додатків від потенційних

загроз. Він дозволив виявляти вразливості на ранніх етапах розробки, що дозволяє виправляти їх, перш ніж вони стануть причиною реальних проблем.

Статистичний аналіз коду можна провести за допомогою таких інструментів як Roslyn або SonarQube [7]. Коротка відомість про ці інструменти:

Roslyn - це платформа для аналізу коду на мові програмування C# та Visual Basic в реальному часі. Вона надає API для розбору, аналізу та модифікації вихідного коду програм. SonarQube - це інструмент для аналізу якості коду, який надає детальну інформацію про потенційні проблеми, вразливості та стилістичні недоліки у вихідному коді програм.

Для аналізу коду за допомогою SonarQube [13], спочатку потрібно налаштувати SonarQube[2] сервер та забезпечити підключення до проекту. Потім можна запустити аналіз за допомогою консольної команди, мавен-плагіну, Gradle або через інтеграцію з вашим середовищем розробки.

```
class Program
{
    static void Main(string[] args)
    {
        // Завантаження файлу з вихідним кодом
        string codeFilePath = @"C:\Projects\MyProject\MyCodeFile.cs"";
        string code = File.ReadAllText(codeFilePath);

        // Створення синтаксичного дерева за допомогою Roslyn
        SyntaxTree syntaxTree = CSharpSyntaxTree.ParseText(code);

        // Створення аналізатора
        var analyzer = new MyAnalyzer();

        // Аналіз синтаксичного дерева
        analyzer.Analyze(syntaxTree);
    }
}

class MyAnalyzer
{
    public void Analyze(SyntaxTree syntaxTree)
    {

        Console.WriteLine("Performing analysis...");
    }
}
```

Він діє як інспектор коду, аналізуючи код, щоб виявити помилки, баги, проблеми, помилки, дублювання та вразливі місця безпеки. «Запахи» коду – це

характеристики коду, які вказують на можливість виникнення проблеми, викликаної кодом у майбутньому.

Код програми системи виявлення вразливості вебдодатка було протестовано за допомогою статистичного аналізу та інструменти Roslyn.

На рисунку 3.4 наведено результат тестування програми, який виведено у формі таблиці.

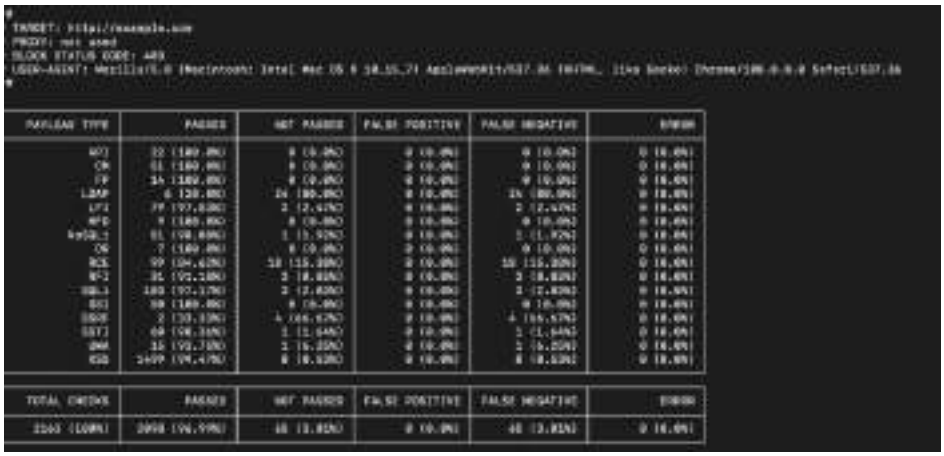


Рисунок 3.4 – Вивід інформації тестування у форматі таблиці

3.6 Валідація та верифікація додатка

Загально визнаний процес валідації — це акт підтвердження, легалізації, визнання, ратифікації, впровадження, підтвердження (підтвердження чинності, законності, юридичної сили); також, акцептування — це процес укладення угоди відповідно до пропозиції іншої сторони.

Верифікація — це підтвердження того, що ствердження є правдивим, у нашому випадку це процес перевірки відповідності програм та їх складових вимогам, які до них пред'являються.[1] Мета верифікації полягає в підтвердженні відповідності програмного забезпечення встановленим вимогам.

Для верифікації WAF було використано інструмент StyleCop [24] , який допоміг виявити прості невідповідності у логіці написаного коду, оптимізувати його та привести до єдиного стандарту. Це дозволило уникнути базових помилок,

які можуть перешкоджати на початковому етапі виконання програми, та забезпечити відповідність коду загальному стандарту. Додатково, верифікація проводилася шляхом регулярного складання діаграм класів кожного модуля, що дозволяло стежити за коректністю архітектури та перевіряти, чи відповідає вона спочатку розробленій архітектурі.

Для валідації WAF застосовувався статистичний аналіз та інструмент Roslyn, а також метод порівняння кінцевого результату написаного коду з діаграмою варіантів використання, яка була розроблена на етапі проектування, а також порівняння з технічним завданням, що було описане на тому ж етапі. Під час валідації додатку перевірялася правильність та відповідність функціоналу, який був заявлений на етапі проектування. Також функціонал проходив перевірку на стійкість та правильність виконання в будь-яких умовах та при будь-яких вхідних даних.

3.7 Аналіз результатів тестування WAF

Після виконання всіх написаних тестів, результати кожного з них було проаналізовано. Компонент перевірки виконується перед усіма іншими функціями WAF, такими як спеціальні правила або керовані правила WAF. Перевірка перевірки блокує неправильно сформовані запити, як-от атаки Shellshock, і запити з певними шаблонами атак у заголовках HTTP до того, як з’явиться будь-яка логіка білого списку. Дії, які виконує компонент перевірки, відображаються в журналі активності, пов’язаному зі службою перевірки, без ідентифікатора правила. Події безпеки, завантажені з API, відображають джерело як перевірку, а дію – як скидання, коли відбувається така поведінка.

Під час тестування кваліфікаційної роботи було проведено аналіз всіх критичних областей додатку, які могли б викликати серйозні проблеми, такі як витік або зміна даних і порушення роботи WAF. Після виявлення вразливих місць

обрано метод статистичного аналізу. Були написані тести та проведене безпосереднє тестування. Результати тестування були ретельно проаналізовані, і вони свідчили про успішне проведення всіх тестів. В результаті цього аналізу можна зробити висновок, що система функціонує коректно і не виявлено непередбачених аномалій у роботі або помилкової обробки вхідних даних.

У четвертому розділі було проаналізовано різні методи тестування та обрано найбільш оптимальний для даної системи. Вибраний метод був докладно протестований, і після цього була здійснена валідація та верифікація платформи. Результати цих процесів показали, що система працює бездоганно, не виявлено жодних аномалій у роботі, і всі модулі ефективно обробляють та приймають дані.

ВИСНОВКИ

У результаті розробки даної кваліфікаційної роботи було створено систему виявлення вразливості вебдодатка. У першому розділі проведено аналіз предметної області, а також розглянуто найпопулярніші аналогічні додатки, платформи та сервіси. Було опрацьовано велику кількість матеріалу, для того щоб реалізувати поставленні задачі. На основі проведених досліджень та розроблених модулів системи виявлення вразливостей вебзастосунку було розроблено систему яка виявляє різноманітні типи вразливостей вебдодатків, такі як SQL-ін'єкції, перехоплення сесій, XSS атаки та інші. При розробці даного застосунку було опрацьовано на прикладах інших систем такого типу та реалізовано учасних методів сканування і аналізу, які дозволяють системі швидко і точно ідентифікувати потенційні загрози без значного втручання оператора.

Для перевірки функціональної частини розробленої системи було проведено тестування, яке підтвердило високу точність і надійність системи, що є критичними параметрами для забезпечення безпеки вебдодатків у сучасному інтернет-середовищі. Розроблена система може слугувати основою для подальших досліджень у напрямку автоматизації виявлення вразливостей вебдодатків, а також для інтеграції з існуючими інструментами розробки та тестування. Застосування розробленої системи дозволяє підвищити рівень безпеки вебдодатків та зменшити ризики їхнього використання для недозволених цілей.

У другому розділі були розроблені основні та допоміжні модулі, спроектовано базу даних, і створено ескізи майбутніх сторінок та компонентів клієнтського додатку. В процесі проектування були обрані відповідні технології для оптимальної реалізації всього задуманого функціоналу.

У третьому розділі була розроблена база даних, реалізовані всі раніше спроектовані модулі та частини системи як на серверній, так і на клієнтській стороні.

У четвертому розділі було описано методи тестування та обрано один із них, який найкраще справився із своєю роботою, а також проведено детальне тестування всіх модулів та підмодулів для перевірки коректності роботи на кожному етапі, а також валідації обробки вхідних даних, включаючи випадки коректних та некоректних даних.

Отже, враховуючи вищевикладене, можна зробити висновок про ефективність та перспективність використання розробленої системи виявлення вразливостей вебзастосунків у практичних умовах.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Babel Ua. URL: https://babel.ua/texts/23307-28-rokiv-tomu-zapustili-pershiy-u-sviti-veb-sayt-mi-pokazali-yak-viglyadav-bi-thebabel-v-1990-roci (дата звернення: 01.05.2024 р.).

2. PortSwigger. URL: https://portswigger.net/burp (дата звернення: 11.05.2024 р.).

3. Wikipedia UA. URL: https://uk.wikipedia.org/wiki/ (дата звернення: 12.05.2024 р.).

4. Microsoft Docs. URL: https://docs.microsoft.com/en-us/ef/core/. (дата звернення: 18.05.2024 р.).

5. C# Corner. URL: https://www.c-sharpcorner.com/article/difference-between-net-framework-and-net-core/ (дата звернення: 18.05.2024 р.).

6. Opentext URL: https://www.opentext.com/what-is/sast (дата звернення: 18.05.2024 р.).

7. My Tech Ramblings URL: https://www.mytechramblings.com/posts/how-to-integrate-your-roslyn-analyzers-with-sonarqube/ (дата звернення: 05.05.2024 р.).

8. Березенко В. В. PR як сфера наукового знання : монографія / за заг. наук. ред. В. М. Манакіна. Запоріжжя : ЗНУ, 2015. 362 с. (дата звернення: 05.05.2024 р.).

9. Бутко М. П., Неживенко А. П., Пепа Т. В. Економічна психологія : навч. посіб. / за ред. М. П. Бутко. Київ : ЦУЛ, 2016. 232 с. (дата звернення: 14.05.2024 р.).

10. Дахно І. І., Алієва-Барановська В.М. Право інтелектуальної власності : навч. посіб. / за ред. І. І. Дахна. Київ : ЦУЛ, 2015. 560 с. (дата звернення: 16.03.2024 р.).

11. Микитів Г. В., Кондратенко Ю. Позатекстові елементи як засіб формування медіакультури читачів науково-популярних журналів. Актуальні проблеми медіаосвіти в Україні та світі : зб. тез доп. міжнар. наук.-практ. конф.,

м. Запоріжжя, 3-4 берез. 2016 р. Запоріжжя, 2016. С. 50–53. (дата звернення: 17.04.2024 р.).

12. C# Tutorial URL: https://www.javatpoint.com/c-sharp-tutorial (дата звернення: 13.05.2024 р.).

13. SONARQUBE WALLBOARD URL: https://itbusina.com/sonarqube-wallboard (дата звернення: 18.05.2024 р.).

14. TechTarget. URL: https://www.techtarget.com/definition/integration-testing/ (дата звернення: 29.05.2024).

15. Can I use. URL: https://caniuse.com/documents/mdn-javascript_builtins_function_apply (дата звернення: 21.05.2024)

16. Руководство по MS SQL Server 2019 [Електронний ресурс] Метанит – Режим доступу URL: https://metanit.com/sql/sqlserver/ (дата звернення: 06.02.2024 р.).

17. Карпова Т.С. Базы данных: модели, разработка, реализация / Т.С. Карпова. СПб: Питер, 2010. 304 с (дата звернення: 06.02.2024 р.).

18. Windows Forms documentation [Електронний ресурс] Microsoft – Режим доступу URL: https://docs.microsoft.com/ruru/dotnet/desktop/winforms/?view=netdesktop-6.0 (дата звернення: 24.02.2024 р.).

19. SQL база даних Для чого призначена база даних? [Електронний ресурс] hosting Ukraine – Режим доступу URL: https://www.ukraine.com.ua/uk/blog/programming/sql-baza-dannih-dlya-chegoprednaznachena-baza-dannih.html (дата звернення: 03.03.2024 р.).

20. UKEY URL: https://ukeywaf.com/baza (дата звернення: 05.03.2024 р.).

21. Synopsys URL: https://www.synopsys.com/glossary/what-is-csrf.html (дата звернення: 18.02.2024 р.).

22. AlibabaClaud URL: https://www.alibabacloud.com/help/en (дата звернення: 18.02.2024 р.).

23. Работа с базами данных в C# и .NET [Електронний ресурс] Метанит Режим доступу URL: https://metanit.com/sharp/ado.php (дата звернення: 18.02.2024 р.).

24. Cod-maze URL: https://code-maze.com/dotnet-stylecop-analyzers implementation/ (дата звернення: 10.02.2024 р.).

25. Corewin URL: https://corewin.ua/blog (дата звернення: 25.05.2024 р.).

26. Воробйов Є. М. Економічна теорія : навч. посібник / Є. М. Воробйов, А. А. Гриценко, В. М. Лісовицький ; за ред. Є. М. Воробйова. – 2- ге вид. — К. ; Х. : Вища школа, 2001. – 704 с. (дата звернення: 19.03.2024 р.).

27. Брильов С. І. Удосконалення інформаційного забезпечення промислових підприємств / С. І. Брильов // Голос України. – 2006. – 12 жовтня. – С. 3. (дата звернення: 07.02.2024 р.).

28. Корпань Я.В. Класифікація загроз інформаційній безпеці в комп'ютерних системах при віддаленій обробці даних. Реєстрація, зберігання і обробка даних. 2015. Т.17. №2. URL: http://dspace.nbuv.gov.ua/bitstream/handle/123456789/ 131565/04-Korpan.pdf?sequence=1. (дата звернення: 01.05.2024 р.).

29. Домарев В.В. Безопасность информационных технологий Методология создания систем защиты. К. : ООО «ТИД «ДС»», 2001. 688 с. (дата звернення: 18.05.2024 р.).

30. Болехівський Н. Полотай О. Класифікація мережевих атак та методи протидії і ззахисту. UR: https://sci.ldubgd.edu.ua/bitstream/handle/123456789/6737/1.pdf?sequence=1&isAllowed=y (дата звернення: 17.05.2024 р.).

31. Lattice Semiconductor. URL: https://latticesemi.com/reference-designs/aesdecryption-documentation (дата звернення: 12.05.2024).

32. Swagger. URL: https://swagger.io/docs/specification/about/ (дата звернення: 12.05.2024).

33. Vue.js. URL: https://vuejs.org/guide/introduction.html (дата звернення: 12.05.2024

ДОДАТОК А

(обов’язковий)

ТЕХНІЧНЕ ЗАВДАННЯ

Введення

Робота виконується в рамках проекту розробки системи виявлення вразливості вебзастосунку «WAF-Security». Технічне завдання розроблено у відповідності до стандарту ГОСТ 19.201–78.

1 Підстава для розробки

Підставою для розробки є «Завдання для кваліфікаційної роботи», затверджене завідувачем кафедри кібербезпеки.

Найменування розробки: Система виявлення вразливості вебзастосунку «WAF-Security»

2 Призначення розробки

2.1 Функціональне призначення

Система виявлення вразливості вебзастосунку «WAF-Security» призначена для виявлення та запобігання атак на вебзастосунки.

2.2 Експлуатаційне призначення

Дана система повинна експлуатуватися на будь-якому пристрої, який має доступ до мережі Інтернет.

3 Вимоги до програми

3.1 Вимоги до функціональних характеристик

Система виявлення вразливості вебзастосунку повинна виконувати такі функції:

- можливість сканування;
- можливість запису результату сканування;

- можливість виявлення вразливостей;
- можливість сканування на конкретні типи атак;
- можливість авторизації;
- можливість висвітлення результатів сканування

3.2 Умови експлуатації та вимоги до технічних засобів

Для успішного користування даною системою користувачу необхідно мати стабільне інтернет з’єднання, додаток веб-браузера, а також його пристрій повинен відповідати наступним мінімальним вимогам:

- будь-яка операційна система;
- будь-який двох ядерний процесор;
- 1 ГБ ОЗП;
- 200 МБ постійної вільної пам’яті;.

3.4 Спеціальні вимоги

Система виявлення вразливості вебзастосунку повинна мати зручний, зрозумілий та привабливий для користувача дизайн.

4 Вимоги до програмної документації

При здачі проекта замовнику, розробник зобов’язається надати наступні документи:

- текстовий вміст програми;
- опис розробленої програми;
- технічне завдання проекту;
- керівництво користувача.

5 Стадії та етапи розробки

Стадії та етапи розробки системи виявлення вразливості вебзастосунку «WEB-Security» подані у таблиці А.1.

Таблиця А.1 – Стадії та етапи розробки проекту

Стадія розробки	Етапи робіт	Зміст робіт
1	2	3
Технічне завдання 02.01.2024 – 31.01.2024	Дослідження предметної області та постановка задачі. Технічне завдання на розробку ПЗ	Дослідження предметної області, в якій планується використання програмного засобу (ПЗ), визначення задач та вимог, розробка технічного завдання
Ескізний проект 01.02.24 – 28.02.24	Проектування ПЗ. Детальний проект ПЗ	Проектування програмного забезпечення

Кінець таблиці А.1

Програмна реалізація 29.02.24 – 19.03.24	Програмна реалізація. Розроблений програмний засіб	Програмна реалізація
Тестування ПЗ 01.03.24 – 10.04.24	Тестування програмного забезпечення. Тест-кейси і тестові сценарії	Тестування програмного забезпечення
Впровадження	Підготовка і передача програми	Підготовка і розгортання програмного забезпечення

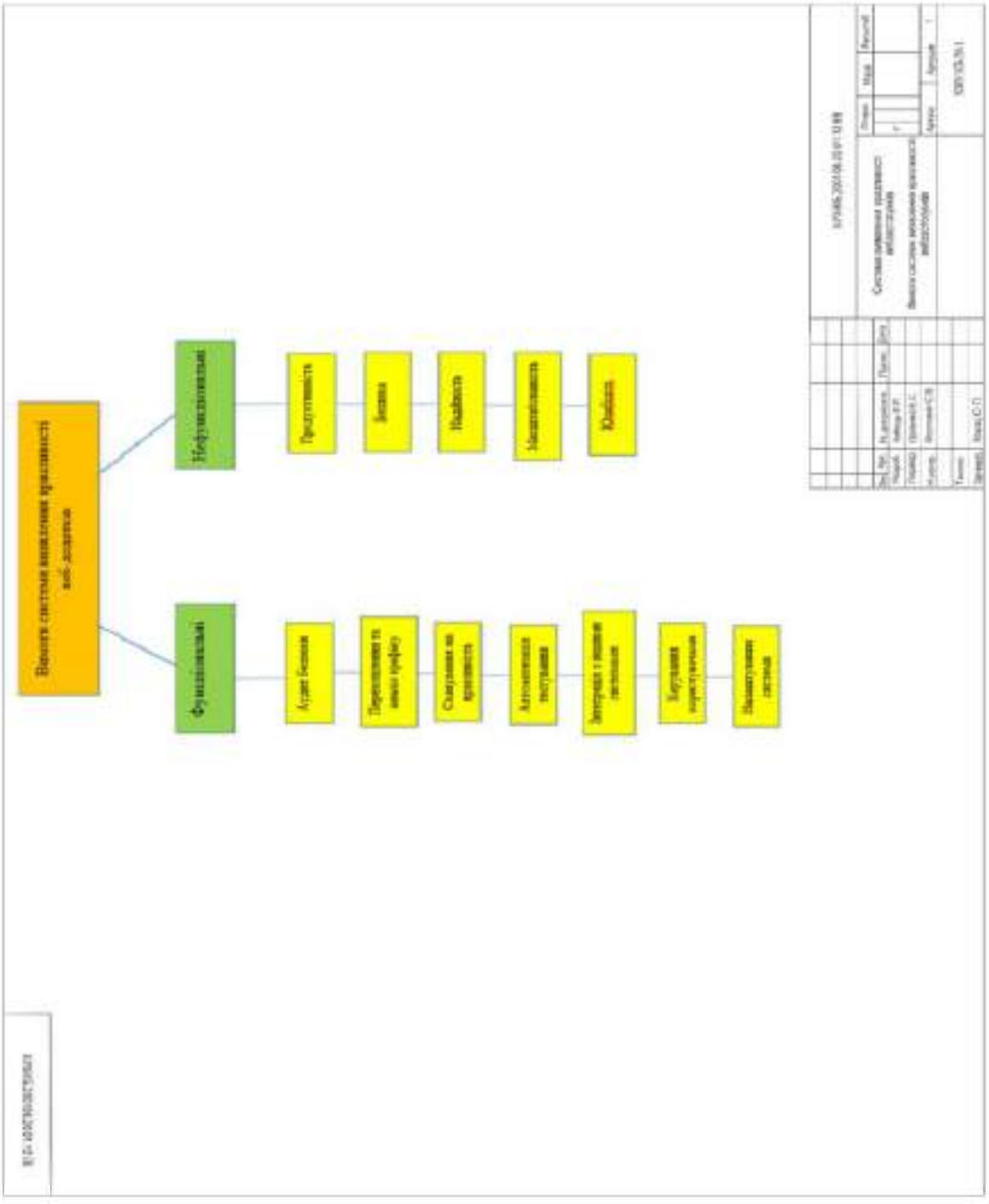
6 Порядок контролю

Контроль над якістю програмного забезпечення здійснюватиметься користувачами системи, які будуть підключені до тестування платформи на етапі тестування ПЗ.

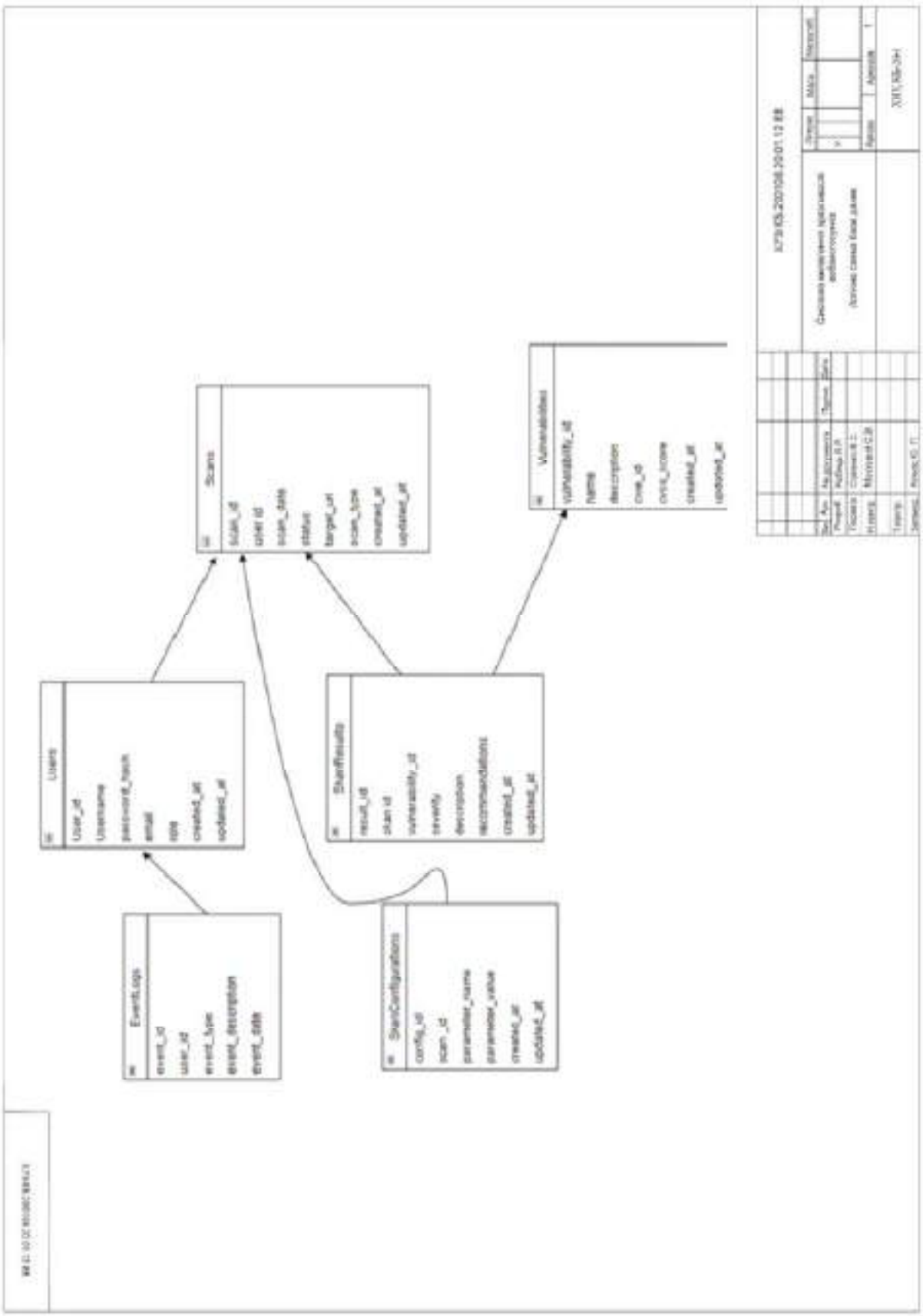
ДОДАТОК Б

(обов’язковий)

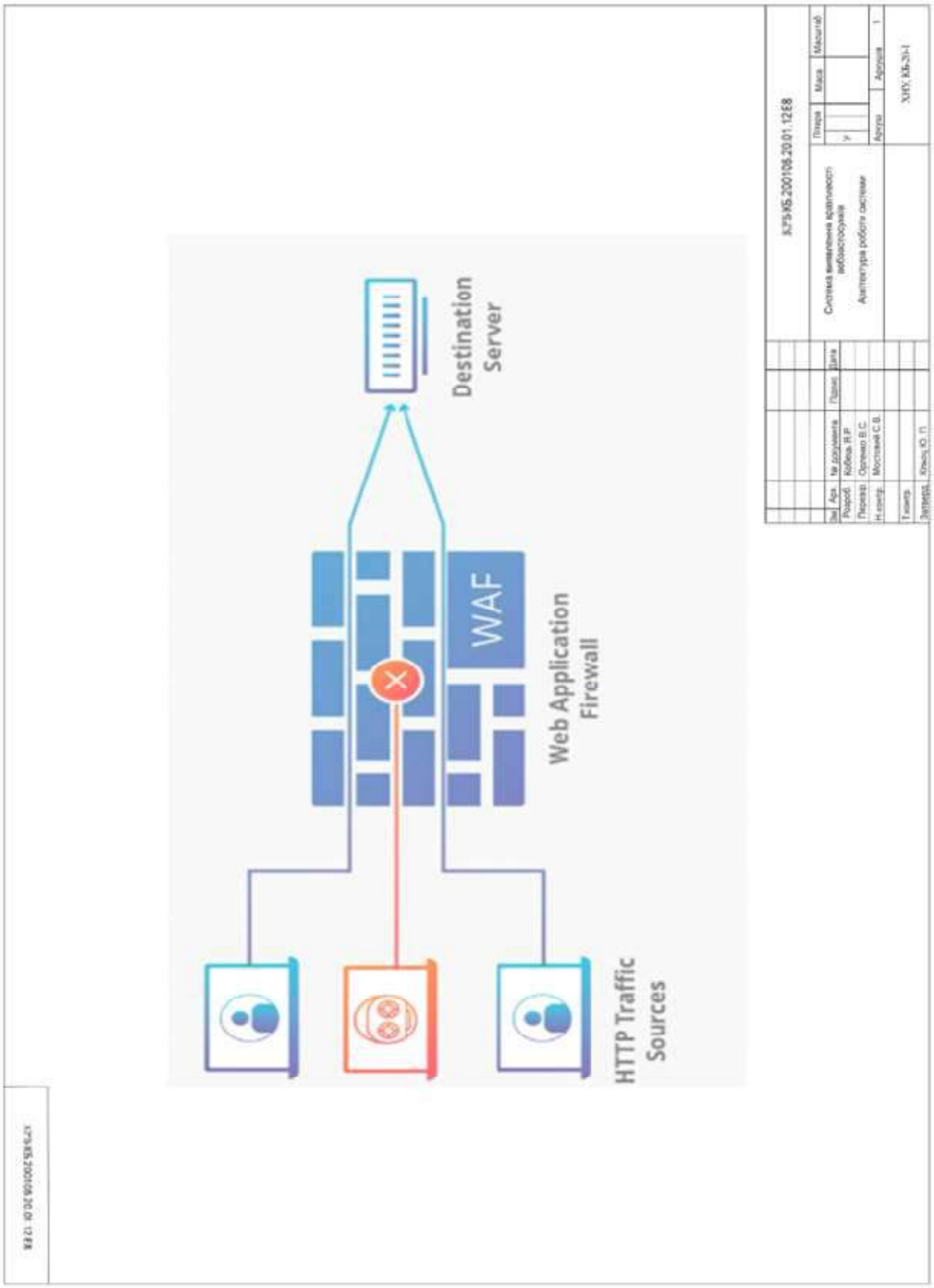
Копії графічної частини



Логічна схема бази даних



Архітектура роботи системи



Завідувачу кафедри кібербезпеки
к.т.н., доц. Кльоцу Ю.П.
Кобець Яни Ростиславівни
ПІБ здобувача вищої освіти

Студентки ФІТ, 4 курсу, групи КБ-20-1

ЗАЯВА

З правилами чинного Положення «Про систему забезпечення академічної доброчесності у Хмельницькому національному університеті» від 31.08.2023, згідно з яким виявлення плагіату є підставою для відмови в допуску кваліфікаційної роботи до захисту та застосування заходів дисциплінарної та академічної відповідальності, ознайомлена. Про використання програмно-технічних засобів для перевірки кваліфікаційних робіт здобувачів вищої освіти на плагіат оповіщена та надаю свою згоду на обробку та збереження університетом моєї роботи в інституційному репозитарії університету.

Також надаю університету право на передачу моєї роботи для обробки та збереження в базах даних програмно-технічних засобів (Unicheck та Anti-Plagiarism) та використання роботи для виявлення плагіату в інших роботах, які перевіряються програмно-технічними засобами та користувачами, що мають доступ до цих програмно-технічних засобів, виключно в обмежених цілях для виявлення плагіату в текстах робіт.

Робота для перевірки університетом надається в друкованому та електронному варіанті. Електронна версія моєї роботи збігається (ідентична) з друкованою.

1.08.2024
дата

підпис

Anti-Plagiarism v-15.257

Максимальне співпадіння з одним документом 1.0%

Словники перевірки: en_US, ru_RU, ua_UA. **Помилок в документах: 9%**

	Документ		Сумарний збіг по Базі Даних	
ID: 132412 Назва: Система виявлення вразливостей вебзастосунків Додано в БД: 2024-06-24 Автора: Кобець Я.Р. Керівники: Орленко В.С. Консультанти: Опоненти:	Символи	Лексеми	Символи	Лексеми
	92805	797	1581 (2%)	21 (3%)

Джерело плагіату

ID	Опис	Наявність плагіату в документі	
		Символи	Лексеми

UNICHECK
by Turnitin

Ім'я користувача:
Кафедра кібербезпеки

ID перевірки:
1016385603

Дата перевірки:
24.06.2024 14:01:28 EEST

Тип перевірки:
Doc vs Internet + Library

Дата звіту:
24.06.2024 14:02:29 EEST

ID користувача:
100008300

Назва документа: **Кобець_на плагіат**

Кількість сторінок: **68** Кількість слів: **13054** Кількість символів: **104876** Розмір файлу: **1.16 MB** ID файлу: **1016196642**

Виявлено модифікації тексту (можуть впливати на відсоток схожості)

4.73%
Схожість

Найбільша схожість: **0.8%** з Інтернет-джерелом (https://wiki.cuspu.edu.ua/index.php/%D0%95%D1%82%D0%B0%D0%BF..

4.04% Джерела з Інтернету	347	Сторінка 70
1.65% Джерела з Бібліотеки	100	Сторінка 72

0% Цитат

Вилучення цитат вимкнене

Вилучення списку бібліографічних посилань вимкнене

0%
Вилучень

Немає вилучених джерел

Модифікації

Виявлено модифікації тексту. Детальна інформація доступна в онлайн-звіті.

Замінені символи	8
Підозріле форматування	11 сторінок

РІШЕННЯ ЕКСПЕРНОЇ КОМІСІЇ

КАФЕДРИ КІБЕРБЕЗПЕКИ

ПРО ДОПУСК КВАЛІФІКАЦІЙНОЇ РОБОТИ ДО ЗАХИСТУ

Підтверджуємо ознайомлення з результатом звіту подібності щодо роботи, генерованого системою виявлення текстових збігів/ідентичності/схожості:

Назва: Система виявлення вразливостей вебзастосунків

Автор: Кобець Яна Ростиславівна

Спеціальність: 125 – Кібербезпека

Освітня програма: освітньо-професійна

Науковий керівник: Орленко Вікторія Сергіївна, канд. техн. наук, доцентка

Після аналізу звіту подібності зроблено такий висновок:

№	Висновок	Позначка про відповідність
1	Запозичення, виявлені в роботі, є законними і не є плагіатом. Робота приймається до захисту.	відповідає
2	Виявлені запозичення не є плагіатом, розміщені в розділах, які не описують безпосередньо авторське дослідження, але кількість цитат перевищує обсяг, виправданий поставленою метою роботи Робота приймається до захисту, але має бути відкоригована. Відкоригований варіант має бути поданий на кафедру за 2 дні до захисту, разом із заявою щодо самостійності виконання письмової роботи та ідентичності друкованої та електронної версії роботи	
3	Виявлені запозичення не є плагіатом, але частково розміщені в розділах, які описують безпосередньо авторське дослідження, а кількість цитат перевищує обсяг, виправданий поставленою метою роботи. В зв'язку з цим мета роботи та поставлені завдання не були досягнені. Робота може бути допущена до захисту (наступного року) після того як буде відкоригована та допрацьована і успішно пройде повторну перевірку на академічний плагіат.	
4	Робота містить навмисні текстові спотворення, передбачувані спроби укриття запозичень або інші прояви академічного плагіату. Робота містить фабрикацію або фальсифікацію даних. Робота не допускається до захисту.	

Підтвердження:

Оригінальність тексту роботи за результатами перевірки системою Unichek складає 95,27%, оригінальність тексту роботи за результатами перевірки системою Anti-Plagiarism v-15.257 складає 99%.

Згідно з Положенням про систему забезпечення академічної доброчесності у ХНУ (https://khmnu.edu.ua/wp-content/uploads/normatyvni-dokumenty/polozhennya/pro-systemu-zabezpechennya-akademichnoyi-dobrochesnosti.pdf, Додаток В) кваліфікаційна робота, виконана за , освітньо-професійною програмою, кількісні показники рівня унікальності тексту у відсотках до загального обсягу матеріалу в якій складає 75-100 %, визнається роботою з високою унікальністю тексту: «Текст вважається унікальним і не потребує додаткових дій щодо запобігання неправомірним запозиченням».

Керівник роботи _____ Вікторія ОРЛЕНКО

Завідувач кафедри кібербезпеки _____ Юрій КЛЬОЦ

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

РЕЦЕНЗІЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ

освітнього ступеня «бакалавр»

Студент Кобець Яна Ростиславівна

Тема Система виявлення вразливості вебзастосунків

Спеціальність 125 – Кібербезпека

Обсяг кваліфікаційної роботи освітньо-кваліфікаційного рівня «бакалавр»:

кількість листів креслень 5; кількість сторінок записки 71.

1. Короткий зміст роботи та прийнятих рішень У результаті розробки даного дипломного проекту було створено систему виявлення вразливості веб-додатка. Було проведено аналіз предметної області, а також розглянуто найпопулярніші аналогічні додатки, платформи та сервіси

2. Висновок про відповідність кваліфікаційної роботи завданню: У дипломному проекті було виконано поставлене завдання як у теоретичній, так і в практичній частині

3. Характеристика виконання кожного розділу роботи, ступінь використання останніх досягнень науки і техніки і передових методів роботи: У вступі роботи наведена загальна характеристика задачі, визначені об'єкт, предмет та методи дослідження, а також сформульована мета. У першому розділі проведено аналіз предметної області, а також розглянуто найпопулярніші аналогічні додатки, платформи та сервіси. У другому розділі були розроблені основні та допоміжні модулі, спроектовано базу даних, і створено ескізи майбутніх сторінок та компонентів клієнтського додатку. В процесі проектування були обрані відповідні технології для оптимальної реалізації всього задуманого функціоналу.У третьому розділі була розроблена база даних, реалізовані всі раніше спроектовані модулі та частини системи як на серверній, так і на клієнтській стороні.

4. Позитивні сторони роботи: Дипломна робота має практичну цінність, через те що було розроблено дієвий застосунок, який виконує поставлені задачі, а саме сканує та вишукує ризики, які в подальшому можуть спричинити витік конфіденційної інформації тощо

5. Негативні сторони роботи: Розроблена система не включає в себе передбачення всіх можливих ризиків нападу та витоку інформації. Даний застосунок, є досить примітивним та дієвим на стандартні можливі атаки

6. Оцінка графічного оформлення та пояснювальної записки роботи: Оформлення графічного матеріалу дипломний проект відповідає тематиці та виконано відповідно до стандартів. Загалом, графічне оформлення відзначається високою якістю, а пояснювальна записка оформлена згідно з нормами.

7. Відгук про роботу в цілому: Дипломний проект заслуговує на високу оцінку, оскільки весь матеріал викладено структуровано, чітко та послідовно. Усі розділи роботи логічно пов'язані, що сприяє кращому розумінню викладеного матеріалу за темою. Графічний матеріал ефективно ілюструє доцільність та ефективність прийнятих рішень для досягнення поставленої мети.

8. Інші зауваження: У списку використаних джерел присутні посилання на популярні ресурси, зокрема Metanit, які не рекомендується використовувати при написанні кваліфікаційних робіт

9. Оцінка кваліфікаційної роботи: Враховуючи всі позитивні та негативні сторони представленої кваліфікаційної роботи, можна зробити висновок, що вона заслуговує оцінки «добре».

РЕЦЕНЗЕНТ (прізвище, ім'я, по батькові, посада, місце роботи) Бармак Олександр Володимирович, д.ф.-м.н, професор, завідувач кафедри комп'ютерних наук

« 19 » червня 2024. (підпис)