Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра захисту інформації

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему:

«Захищена система збирання та аналізування даних для спеціальних задач. Частина 2. Підсистема захисту»

Виконав: студент 2-го курсу, групи 1БС-21м
спеціальності 125 – Кібербезпека
_____Смолявський І.С.

Керівник: к.т.н., ст. викл. каф. ЗІ
_____ Лукічов В.В.

Опонент: к.т.н., доц., доц. каф . ПЗ
_____Майданюк В. П.

«20» грудня 2022 р.

Допущено до захисту

Завідувач кафедри ЗІ
д.т.н., проф.
_____ Лужецький В.А.
«20» грудня 2022 р.

Вінниця ВНТУ – 2022 рік

Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра захисту інформації
Рівень вищої освіти II-й (магістерський)
Галузь знань 12 Інформаційні технології
Спеціальність 125 Кібербезпека
Освітня програма – Безпека інформаційних і комунікаційних систем

ЗАТВЕРДЖУЮ
Завідувач кафедри ЗІ,
д.т.н., проф.
_____ В.А. Лужецький
«15» Вересня 2022 року

З А В Д А Н Н Я
НА МАГІСТЕРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

Смолявському Іллі Сергійовичу

1. Тема роботи: «Захищена система збирання та аналізування даних для спеціальних задач. Частина 2. Підсистема захисту»,
керівник роботи: Лукічов Віталій Володимирович, к.т.н., ст. викл. каф. ЗІ,
затверджені наказом ректора ВНТУ від 14 вересня 2022 року №203.
2. Строк подання студентом роботи 19 грудня 2022 р.
3. Вихідні дані до роботи:
   - дані зібрані з інтернет-форм, фактори ризиків ІБ;
   - спосіб реалізації – веб-ресурс, модулі автентифікації, збирання та аналізування даних, модуль оцінювання ризиків інформаційної безпеки.
4. Зміст текстової частини: Вступ. 1. Аналіз інформаційних джерел. 2. Розробка методу захисту веб-застосунків. 3. Розробка програмного додатку. 4. Експериментальне дослідження. 5. Економічна частина. Висновки. Список використаних джерел. Додатки.
5. Перелік ілюстративного матеріалу: Порівняння аналогів систем для пошуку атак(плакат, А4). Узагальнений алгоритм захисту від загроз веб-застсунків(плакат, А4). Узагальнена структура бази даних(плакат, А4). Результати тестування (плакат, А4). Узагальнена структура бази даних(плакат, А4). Результати тестування (плакат, А4).

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанти	Підпис, дата	
		завдання видав	виконання прийняв
1	Лукічов В.В., к.т.н., старш. викл. каф. ЗІ		
2	Лукічов В.В., к.т.н., старш. викл. каф. ЗІ		
3	Лукічов В.В., к.т.н., старш. викл. каф. ЗІ		
4	Лукічов В.В., к.т.н., старш. викл. каф. ЗІ		
5	Лесько О.Й., к.е.н., професор каф. ЕПВМ		

7. Дата видачі завдання 1 вересня 2022 року

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів магістерської кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Аналіз завдання. Вступ	01.09.2022 – 06.09.2022	
2	Розробка технічного завдання	07.09.2022 – 14.09.2022	
3	Аналіз інформаційних джерел за напрямком магістерської кваліфікаційної роботи	15.09.2022 – 25.09.2022	
4	Розробка рішень	25.09.2022 – 05.10.2022	
5	Практична реалізація, моделювання, експериментування, результати	06.10.2022 – 26.10.2022	
6	Розробка розділу економічного обґрунтування доцільності розробки	26.10.2022 – 18.11.2022	
7	Аналіз виконання ТЗ, висновки	19.11.2022 – 25.11.2022	
8	Оформлення пояснювальної записки	26.11.2022 – 30.11.2022	
9	Попередній захист та доопрацювання МКР	07.12.2022	
10	Представлення МКР до захисту	19.12.2022	
11	Захист МКР	22.12.2022	

Студент _____ І. С. Смолявський

Керівник роботи _____ В.В. Лукічов

АНОТАЦІЯ

УДК 004.056

Смолявський І.С. Захищена система збирання та аналізування даних для спеціальних задач. Частина 2. Підсистема захисту. Комплексна магістерська кваліфікаційна робота зі спеціальності 125 – Кібербезпека, освітня програма – Безпека інформаційних і комунікаційних систем. Вінниця: ВНТУ, 2022. 91 с.

На укр. мові. Бібліогр.: 14 назв; рис.: 37; табл. 15.

Комплексна магістерська кваліфікаційна робота захисту системи збирання та аналізування даних для спеціальних задач, шляхом розробки відповідного методу та алгоритму. Для успішної розробки програмного застосунку проведено аналіз та дослідження наявних аналогів програмних реалізацій систем знаходження та виявлення, а також захисту даних від найпоширеніших кіберзагроз. Розроблено власний комбінований метод виявлення та розпізнавання кіберзагроз, шляхом послідовних дій які передбачають захист від найпоширеніших загроз веб- застосунків Під час роботи обґрунтовано вибір власних методів та обгрунтован середовище розробки застоснку, розроблено ряд схем і алгоритмів, здійснено програмну реалізацію. Засіб перевірено на коректність роботи.

Графічна частина складається з 6 плакатів з демонстрацією схеми алгоритму роботи системи та прикладами її використання.

В економічному розділі оцінено витрати на розробку.

Ключові слова: захищена система, комбіновий метод, веб-додаток, база даних, інформаційна безпека.

ABSTRACT

Smolyavsky I.S. Secure data collection and analysis system for special tasks. Part 2. Protection subsystem. Comprehensive master's qualification work on specialty 125 – Cybersecurity, educational program – Security of information and communication systems. Vinnytsia: VNTU, 2022. 91 p.

In Ukrainian speech Bibliography: 14 titles; Fig.: 37; table 15.

Comprehensive master's qualification work on the protection of the data collection and analysis system for special tasks, by developing the appropriate method and algorithm. For the successful development of the software application, an analysis and research of existing analogues of software implementations of detection and detection systems, as well as data protection from the most common cyber threats, was carried out. An own combined method of detection and recognition of cyber threats was developed, through sequential actions that provide protection against the most common web application threats. During the work, the choice of own methods and the application development environment were justified, a number of schemes and algorithms were developed, and software implementation was carried out. The tool has been checked for correct operation.

The graphic part consists of 6 posters with a demonstration of the algorithm scheme of the system and examples of its use.

Development costs are estimated in the economic section.

Keywords: secure system, combined method, web application, database, information security.

ЗМІСТ

ВСТУП

Захист даних у комп'ютерних мережах стає однією з найвідкритіших проблем у сучасних інформаційно-обчислювальних системах. На сьогоднішній день сформульовано три базові принципи інформаційної безпеки, завданням якої є забезпечення:

- цілісність даних – захист від збоїв, що ведуть до втрати інформації або її знищення;
- конфіденційність інформації;
- доступність інформації для авторизованих користувачів.

Розглядаючи проблеми, пов'язані із захистом даних у мережі, виникає питання про класифікацію збоїв та несанкціонованість доступу, що веде до втрати або небажаної зміни даних. Це можуть бути збої обладнання або ж найпоширеніші види атак, втрати інформації (через інфікування комп'ютерними вірусами, неправильне зберігання архівних даних, порушення прав доступу до даних), некоректна робота користувачів та обслуговуючого персоналу. Перелічені порушення роботи у мережі викликали необхідність створення різних видів захисту.

З огляду на вищевказане тема роботи присвячена вдосконаленню методу та засобів захисту інформації від кібервпливів та виявленню кіберзагроз на початкових рівнях в комп’ютерних системах та мережах об’єктів захисту.

Отже **актуальною** є розробка власного методу захисту веб-застосунків та програмного засобу для його реалізації. Також розроблений комбінований метод розпізнавання кіберзагроз має змогу бути використованим в системі захищеній системі збирання та аналізування даних.

Об’єктом дослідження є процес забезпечення безпеки системи збирання та аналізування даних для спеціальних задач.

Предмет дослідження – модуль підсистеми захисту системи збирання та аналізування даних для спеціальних задач.

Метою комплексної магістерської кваліфікаційної роботи є покращення безпеки системи збирання та аналізування даних для спеціальних задач.

Для досягнення мети потрібно виконати наступні завдання:

- виконати аналіз засобів для виявлення вторгнень;
- проаналізувати відомі методи оцінки розпізнавання кіберзагроз;
- розробити модель захисту системи у вигляді веб-додатку;
- розробити захищенну систему збирання та аналізування даних.

Наукова новизна магістерської роботи полягає в тому, що:

– удосконалено комбінований метод захисту веб-застосунківа а також розроблений комбінований метод розпізнавання кіберзагроз з подальшим розвитком.

Публікації результатів магістерської кваліфікаційної роботи. Результати магістерської роботи доповідалися на таких конференції "Контроль і управління в складних системах (КУСС-2022)" ВНТУ місто Вінниця, 15-17 листопада 2022 р.

1 АНАЛІЗ ІНФОРМАЦІЙНИХ ДЖЕРЕЛ

1.1 Науково-технічне обґрунтування розробки захищеної системи збирання та аналізування даних для спеціальних задач

У процесі розвитку інформаційних технологій паралельно розвивається і сфера інформаційної безпеки. За останні десятиліття десятки різних методів і практик захисту інформації були розроблені та об’єднані в системи управління інформаційною безпекою. Використовуючи ці практики, професіонали з інформаційної безпеки можуть ефективно реагувати та протистояти останнім загрозам, що виникають на інформаційній арені. Одним із таких методів є міжнародний стандарт інформаційної безпеки ISO/IEC 27001. Він включає передові організаційні практики для управління інформаційною безпекою та найкращі способи боротьби з ризиками. Запровадивши цей стандарт у національну систему інформаційної безпеки, держава зможе максимально швидко реагувати на різні загрози в інформаційному середовищі. Відповідність цьому стандарту є великим кроком до визнання перед міжнародним співтовариством, а також покращує економічні показники країни.Виходячи з цього теми роботи є актуальними на сьогоднішній день. Мета роботи – ознайомлення зі стандартом ISO/IEC 27001 та визначити її роль у світовій системі захисту інформації та системі ІБ України. Дослідження проблем впровадження ISO/IEC 27001 в українську систему.

На сучасному етапі розвитку суспільства різноманітні послуги надаються за допомогою Інтернету та комп’ютерних технологій, це стало можливим як технічно, так і економічно, в тому числі психологічно.

Він приносить прибуток лише в останній рік цього століття. Це пов'язано з появою нових форм передачі та обробки даних із впровадженням високоефективне і відносно недороге обладнання, поява багатьох організацій, які надають технічну підтримку розробка модульних елементів для Інтернет-контенту тощо.

У той же час, основна проблема для ефективних і успішних пологів сервіс– це завдання проектування та подальшого впровадження професійна інформаційна система, яка може вирішити різні завдання. Одна з найпопулярніших систем обміну повідомленнями.

Інформацією для широкого кола користувачів останнім часом є веб-сайт, що містить дані з різних галузей. простота і простота використання сайту та його універсальність роблять сайт популярним у всіх сферах людської діяльності [1].

Усі державні організації, залучені до суспільства сьогодні проблеми, що передбачають використання інформаційних технологій, особливо веб-сайти. Вони формують і задовольняють інформаційні потреби надання інформації про потреби людей і суспільні процеси, поліпшити умови життя всіх членів суспільства та підвищити якість послуг, що надаються інформаційних послуг, а в цілому – забезпечити радикальні покращення соціальна діяльність і життя людини.

Добре проведена атака створює загрозу для роботи сайту, і логічним результатом є фінансова та репутаційна шкода. Злочинці можуть використовувати цей сайт як основний ресурс для надсилання спаму або атак DoS для атак на інші ресурси. В результаті ваш сайт буде заблоковано пошуковими системами та браузерами, і ви втратите користувачів. Крім того, атаки можуть бути спрямовані на подальше «зараження» користувачів сайту, наприклад, за допомогою так званих пакетів експлойтів (використання вразливостей у браузерах та їх компонентах).

Для досягнення цієї мети кожній організації необхідно розробити систему управління інформаційною безпекою (СУІБ) відповідно до рівня інформації, але має можливість захистити інформаційні активи.

В організації, існування якої значною мірою залежить від інформаційних технологій, можна використовувати всі інструменти для захисту даних. Однак інформаційна безпека має важливе значення для споживачів, ділових партнерів, інших організацій та урядів. У зв’язку з цим20 кожна організація повинна

прагнути запровадити певну стратегію та систему безпеки для захисту цінної інформації.

СУІБ – це цілі та вимоги організації, вимоги безпеки, які використовуються в її процедурах, а також розміри та структура її організації.

Як частина ISO/IEC Підкомітет 27 відповідає за розробку сімейства міжнародних стандартів для управління інформаційними системами, звідси нумерація цієї сімейства стандартів за допомогою серії послідовних номерів, починаючи з 27000 (27k).Прийнята схема. Оскільки 25 вересня 2013 року було опубліковано нові версії стандартів ISO/IEC 27001 та 27002, серію стандартів ISO/IEC 27k (управління ІС) було повністю об’єднано з серією стандартів ISO/IEC 20k (керування ІТ-послугами). Інтегрований.). Усі терміни з ISO/IEC 27001 були перенесені до ISO/IEC 27000. ISO/IEC 27000 визначає загальну номенклатуру в сімействі стандартів ISO/IEC 27k. Інформаційна безпека пройшла довгий шлях за останні півстоліття. Зародившись жартом серед колег у 1960-х роках, технологічний прогрес, який постійно змінюється, зробив інформаційну безпеку головною проблемою та потребою сучасності.

Основні загрози захисту інформації в бездротових мережах полягають в наступному:

- несанкціоноване перехоплення даних, переданих по бездротових мережах зв'язку;
- мережеві атаки, спрямовані на обладнання бездротових мереж зв'язку;
- атаки, спрямовані на отримання несанкціонованого доступу до ресурсів системи через бездротову мережу зв'язку;
- несанкціоноване спотворення даних, що передаються через бездротову мережу зв'язку.

Усі зловмисник мають якісь певні цілі та намагається проникнути до мережі безпровідного доступу :

- викрадення чи зміна персональних даних користувача в мережі;
- доступ до каналу інтернет через комп’ютер жертви;

– повне руйнування локальної мережі.

Тобто зловмисник може якимось чином вплинути на цілісність інформації в мережі, її доступність і конфіденційність або просто незаконно використовувати інтернет-канали у власних цілях без авторизації вразливих користувачів. Основними недоліками бездротових мереж, які роблять їх уразливими до атак, є старе обладнання зі старими конфігураціями та неправильне налаштування існуючого обладнання. Чим більше функцій має маршрутизатор, тим більша ймовірність виявлення невиправлених уразливостей.

А деякі заявлені виробником функції, наприклад, можливість захисту від несанкціонованого доступу, лише ускладнюють життя користувачам. Однією з таких функцій є білий список MAC. Зловмиснику потрібно лише кілька хвилин, щоб замаскувати MAC-адресу білою. Якщо ви звичайний користувач бездротового зв'язку і бажаєте підключити новий пристрій, вам доведеться звертатися до системного адміністратора або кожного разу змінювати його [1].

Власні налаштуванняточки доступу та глобальна статистика безпеки точки доступу показані на (рис. 1.1)

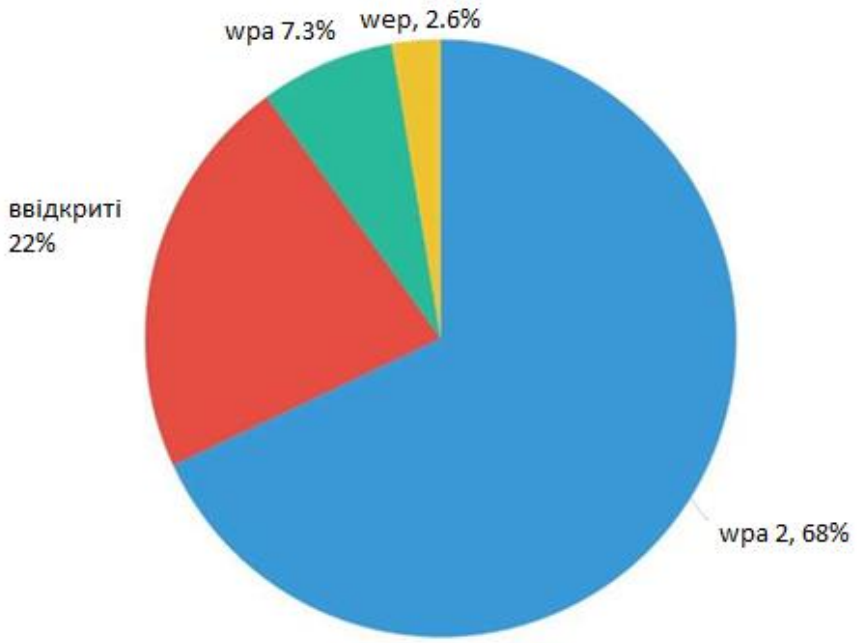


Рисунок 1.1 – Світова статистика захищеності точок доступу

Існуючі системи виявлення вторгнень в бездротових мережах орієнтовані на аналіз протоколів бездротового зв'язку сімейства IEEE 802.11, ідентифікацію та аналіз підозрілої активності.

Розрізняють кілька основних видів атак, які загрожують безпеці бездротових комп'ютерних мереж:

- «Людина посередині»;
- DDoS-атаки;
- помилкова точка доступу;
- атаки на мережеве обладнання.

Статистика показує, що Wi-Fi мережі найчастіше зламують за допомогою дистрибутивів Kali-Linux. Kali-Linux містить низку програмного забезпечення та утиліт, які можна використовувати для тестування на проникнення в мережі Wi-Fi тощо.

Інформаційна безпека пройшла довгий шлях за останні півстоліття. Інформаційна безпека, яка виникла в 1960-х роках з жартів між колегами, і невпинного технологічного прогресу, що відбувся протягом наступних кількох років, стала головною проблемою, а отже, сучасною потребою.

1.2 Аналіз основних вразливостей сучасних веб-ресурсів

Для того щоб логічно оцінити та провести аналіз засобів захисту потрібно добре розуміти основні вразливості. Найпоширенішими на сьогодні веб-серверами заобсягом використання на працюючих сайтах та сервісах є такі [4]:

1) Apache HTTP-server – близько 50% від загальної кількості;

2) nginx HTTP-server – близько 25% від загальної кількості;

3) IIS (Microsoft) – близько 15% від загальної кількості.

Переглядаючи ці вразливості веб-сервера, ми бачимо, що найпоширенішою вразливістю є так звана «відмова в обслуговуванні». Його відносно легко можна було придбати на чорному ринку та відправити на будь-який мережевий ресурс. Тому були розроблені контрзаходи саме для запобігання цій атаці.

Більшість об’єктів цієї атаки – сайти, створені для заробітку. Для інтернет-магазину просте закриття служби призведе до зменшення кількості замовлень і меншого прибутку. Цей тип атаки зазвичай є атакою конкурента. І подібні «атаки» можуть відбуватися з ідеологічних мотивів. Хакери можуть вимагати гроші, щоб відновити ваш сайт, але це рідкість.

DDoS-атака — це розподілена атака на відмову в обслуговуванні на різноманітні інтернет-сервіси. Якщо така атака вдасться сервер перестає відповідати на законні запити користувачів. Масована DDoS атака

Веб-сайти урядів і органів влади, веб-сайти великих IT-компаній, Amazon, Yahoo, такі як Microsoft Ці потужні компанії з величезними ресурсами не завжди можуть впоратися з атаками та відбити їх. За даними [1], кількість DDoS-атаки постійно зростають. Світові лідери в інформаційній безпеці показують необхідність виявлення DDoS-атак і протидії їм як основне завдання досліджень. Це є основною причиною для розроблення та впровадження заходів захисту від проблема DDoS-атаки актуальна [2].

Поточний один з найпоширеніших видів DDoS-атак. Існують атаки, засновані на віддзеркаленні та посиленні шкідливого трафіку. При розробці нових універсальних методів захисту від таких атак типом пріоритетного завдання є аналіз протоколів, які можуть бути використані для:здійснення таких атак.

Крім того, аналіз існуючих методів захисту атаки, які відображають трафік за допомогою різних протоколів. Представлені результати аналізу та висновки на його основі.

Особливості та недоліки традиційних методів захисту. Першим недоліком традиційних систем захисту від DoS є те, що вразливість залежить від можливостей апаратного забезпечення, на якому розгорнуто ці веб-сервери, а не від правильно налаштованого програмного забезпечення. Завжди обмежений. Іншими словами, «розумний» фільтр може стати основним споживачем системних ресурсів, якщо він перевіряє всі вхідні з'єднання під час атаки, викликаючи відмову в обслуговуванні на вашому сервері.. Другий недолік полягає у неможливості відрізнити «розумних» ботів від справжніх користувачів без складних обчислень і витрат ресурсів серверу. Схема захисту від DoS-атак стандартними методами наведена на (рис. 1.2)

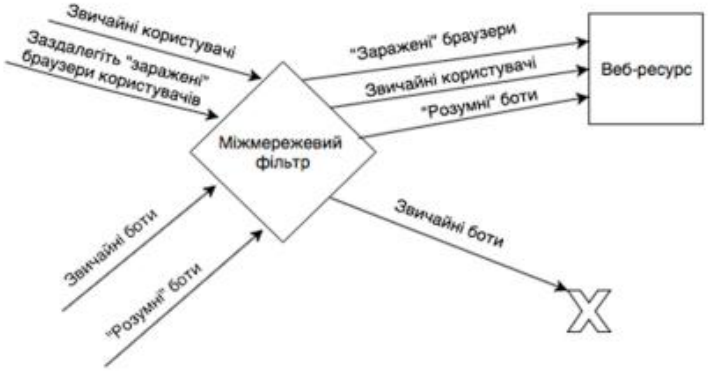


Рисунок 1.2 – Захист від DoS-атак стандартними методами

Як відомо, кожна хвилина роботи комерційного веб-сайту чи сервісу коштує вашій компанії певних грошей, а кожна хвилина простою в ситуації відмови в обслуговуванні коштує вам дуже дорого. Послуги і, як наслідок, не генерують товарні та грошові потоки. З цих причин ця робота пропонує методи боротьби з DoS-атаками з метою підтримки нормальної лояльності користувачів. Дохід залежить від кількості особливо ентузіазмів і лояльних клієнтів. І чим більше користувачів вам довірятиме, тим кращим буде ваш бізнес у довгостроковій перспективі. Щоб надати користувачам пріоритетний доступ до сайту чи послуг [2]. По відношенню до стандартних методів захисту, розроблений метод дає можливість відсіювати «розумних» ботів, які нічим не

відрізняються від звичайних користувачів, тим самим зменшуючи споживання ресурсів сервера.Легко зрозуміти, що запропонований метод розробленого методу боротьби з відмовою -атаки служби складаються з:

- Ідентифікація користувача за допомогою цифрового відбитку браузера.
- Запис ідентифікатора на комп’ютер користувача для полегшення роботи фільтру під час атаки.
- Присвоєння кожному ідентифікатору певного параметру «рейтингу лояльності» (рівня довіри до цього користувача) та запис цього значення на сервер.
- Використання параметру «рейтингу лояльності» під час перевищення ліміту запитів на сайт для аутентифікації лояльного користувача та для його відмінності від ботів.
- Надання доступу до сайту користувачам з високим «рейтингом лояльності» і навпаки, відмова у обслуговуванні користувачів з меншим «рейтингом лояльності», за допомогою фільтру між клієнтом і сервером.

Ідентифікація за допомогою цифрового відбитка браузера Суть технології зняття відбитків браузера полягає в тому, що код запитує браузер користувача всі унікальні налаштування і дані цього браузера, цієї системи і вашого комп'ютера. Для ідентифікації можна використовувати, наприклад, такі дані [2]: часовий пояс, розмір екрану, масив, глибина кольору; системні шрифти; SessionStorage, LocalStorage, IndexedDB, OpenDatabase та інші стандартні технології HTML5. Налаштування процесора doNotTrack, клас процесора, тип платформи та інші дані, пов'язані з користувачем та платформою. Інформація про плагіни. Об'єднайте всі ці вхідні дані в рядок та передайте його на вхід хеш-функції. Хеш-функція використовує його і перетворює у вихідний файл на 32-бітове число. Це буде ваш ідентифікатор користувача. Загалом, можна зібрати більше або менше даних для ідентифікації, роблячи здогад клієнта більш точною

або менш точною, але чим вона точніша, тим більше ресурсів сервера потрібно виділити на цю операцію.

Формування "лояльності" користувачів. Для формування рейтингу лояльності значення параметра (наприклад, час, проведений на сайті або веб-сервісі) розраховується для кожного користувача з ідентифікатором, а потім цей параметр порівнюється з тимчасовими параметрами інших користувачів. сайт або веб-додаток. Ви також можете порівняти отримані параметри із середніми значеннями всіх користувачів. Порівняння результатів може включати кілька станів, таких як «невідомий користувач», «відомий користувач», «постійний користувач» і т. д. Запишіть його на сервер і зв'яжіть з ідентифікатором. Залежно від деталей сайту або сервісу, що розглядаються в індивідуальному порядку, тимчасовий параметр може бути замінений іншим, що відображає лояльність користувача. Наприклад, кількість відвідуваних сторінок веб-сайту або кількість придбаних товарів. Користувачі, які пройшли процес реєстрації на Сайті та надали особисту інформацію, можуть вважатися беззастережно лояльними.

Реєстрація ідентифікаторів за клієнта. Для автентифікації користувача при перевищенні ліміту запитів до сервера вам необхідно довести, що ви не належите до бота, що атакує сервер. Файли cookie із раніше записаними ідентифікаторами (цифровими відбитками браузера) можна використовувати для зниження навантаження на фільтри. Справжні елементи, які зберігають інформацію у файлі cookie, є елементами, які не можна видалити. Ви можете використовувати evercookie для цього. Він не тільки зберігає дані, такі як файли cookie у своєму сховищі, але й використовує всі доступні репозиторії сучасних веб-браузерів. Звичайний користувач з поверхневими знаннями не може видалити файли cookie. Це пов'язано з тим, що для видалення файлів cookie потрібно перейти в 6-8 місць на жорсткому диску і виконати ряд дій [4].

Включення операцій сайту перед потенційними DoS-атак Для реалізації фільтрації користувачів ви можете використовувати проксі-сервери, такі як nginx або lighttpd. Режим фільтрації активується при перевищенні кількості запитів на

сервер. У разі перевищення ліміту запитів ініціюється режим роботи «потенційна DoS-атака». Алгоритм роботи фільтру під час цього режиму:

- Підключитися до сервера, клієнт надсилає cookie з HTTP-запитом. Фільтр зчитує ідентифікатор, знаходить відповідний параметр лояльності і перенаправляє клієнта на сервер, якщо параметр відповідає певним критеріям.
- Без файлів cookie фільтр використовує цифровий відбиток пальця для ідентифікації вашого браузера та пошуку відповідності у базі даних відомих користувачів. Якщо збіг знайдено і параметри «Рейтинг доступності» вірні, перенаправляємо клієнта на сервер.
- В іншому випадку проксі-сервер відхилятиме запити на доступ до сервера доти, доки кількість запитів до сервера не повернеться до норми (нижче за максимальний ліміт запитів). Те, як саме користувач буде заборонено або дозволено на сайті, залежить від індивідуально обраної технології фільтрації (інтернет-екран).

Завдяки цьому алгоритму лояльні, постійні користувачі вашого сайту або веб-застосунки можуть безперешкодно користуватися вашим веб-сервером, але боти та ненадійні користувачі не можуть отримати доступ до вашого сайту, що знижує навантаження на ваш сервер.

Захист від DoS-атак за допомогою пропонованого методу у поєднанні зі стандартним. Завдяки цьому алгоритму лояльні, постійні користувачі вашого сайту або веб-застосунки можуть безперешкодно користуватися вашим веб-сервером, але боти та ненадійні користувачі не можуть отримати доступ до вашого сайту, що знижує навантаження на ваш сервер.

Захист від DoS-атак за допомогою пропонованого методу у поєднанні зі стандартним (рис. 1.3)

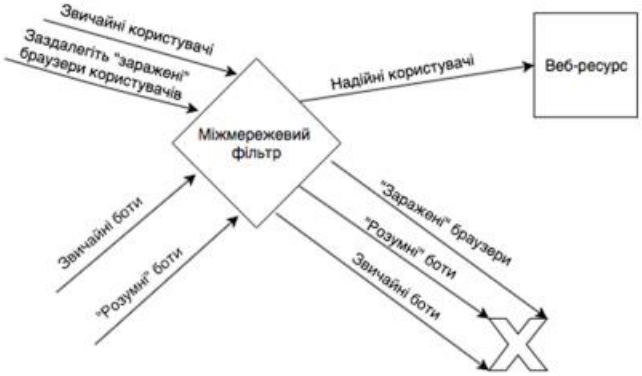


Рисунок 1.3 – Захист від DoS-атак з використанням запропонованого методу у поєднанні зі стандартними

Це робить DoS-атаки неефективними, оскільки ваш сайт чи сервіс ніколи не втрачає своїх постійних, лояльних користувачів. Таким чином, ви не втрачаєте більшу частину цієї вигоди, як у випадку із загальною відмовою в обслуговуванні для всіх користувачів.

За результатами є способи боротьби з DoSatak, які враховують інтереси власників сайтів та їх постійних користувачів. Запропоновано простий алгоритм дій щодо підготовки заходів протидії DoS-атаці та алгоритм дій за можливої DoS-атаки.

Пропонована методологія настільки гнучка і масштабована, що її можна використовувати для захисту широкого профілю сайтів та сервісів незалежно від деталей та обсягу їх відвідувань.

1.3 Аналіз засобів-аналогів для виявлення вторгнень

Бездротові корпоративні мережі постійно зазнають атак зловмисників через широкомовний характер радіохвиль, які не обмежуються стінами будівель. Тому питанням безпеки бездротової мережі слід приділити особливу увагу [6].

Існуючі системи виявлення вторгнень бездротові мережі зосереджені на аналізі сімейства протоколів бездротового зв'язку IEEE 802.11.

Для вирішення проблеми безпеки бездротових локальних мереж багато організацій розгорнули або планують розгорнути систему запобігання вторгненням у бездротову мережу (WIPS — Wireless Intrusion Prevention System). Вони призначені для моніторингу активності бездротової мережі та виявлення/запобігання спробам вторгнення у внутрішні та зовнішні мережі. На основі аналізу, заснованого на канальному та фізичному рівнях мережної моделі OSI, цей інструмент допомагає організаціям краще ідентифікувати та захищати свої мережі від шахрайських точок доступу, бездротових атак та атак типу «відмова в обслуговуванні».

Основні загрози, яким може піддаватися корпоративна бездротова мережа:

- Безпровідний вплив – з'єднання між користувачем і точкою доступу обривається за допомогою відправки повідомлення про дисоціації (роз'єднання), після чого точка доступу відмовляє у відновленні з'єднання;
- мережевий вплив – система передає комутатора команду блокувати з'єднання з даним користувачем мережі по порту або MAC-адресою.

Крім того, деякі системи можуть визначити фізичне розташування джерела виявленої загрози за допомогою методу тріангуляції.

Ця функція виявляє наступні вразливості:

- Ненадійні або задані за замовчуванням паролі (для адміністрування маршрутизатора та мережі Wi-Fi).
- Уразливості вбудованого ПЗ (для найбільш відомих постачальників).
- Бездротові мережі без шифрування і захисту.
- Злам DNS (для пристроїв і маршрутизаторів).
- Відкриті порти мережі (для віддаленого доступу, Telnet і так далі). Результати сканування містять:
- IoT прилади в мережі.
- Данні маршрутизатора (IP адреса, MAC адреса, його постач, модель, SSIDта DNS)[7].

За допомогою програми для Android ви можете перевірити не тільки безпеку вашої точки доступу, а й підключених пристроїв.

Однак усі ці функції працюють лише при ручному запуску, а в контрольованому режимі вони не працюють навіть у комерційній версії [9].

Утиліта waidps може використовуватися для виявлення аномалій бездротового ефіру в домашніх умовах. WAIDPS – це програма з відкритим вихідним кодом, написана на Python і працює в середовищі Linux. Точні залежності не вказані, але при запуску з Kali програма створить/скопіює необхідні бази даних та буде готова до використання. Іншими словами, Kali Linux включає всі компоненти, необхідні для цієї програми.

Це багатоцільовий інструмент, створений для аудиту мереж (тестування на проникнення), виявлення бездротових вторгнень (атаки WEP/WPA/WPS) та запобігання вторгненням (припинення зв'язку між станціями та точками доступу) [10].

Крім того, програма збирає всю інформацію про WiFi у вашому районі та зберігає її в базі даних. Це корисно при аудиті вашої мережі. Якщо увімкнено точку доступу з MAC-фільтром або Прихованим SSID.

Програма сама викликає необхідні інтерфейси та читає для моніторингу трансляції. Цікавою особливістю є те, що програму можна використовувати не лише для виявлення атак, але й для запуску бездротових мереж.

Для роботи програми необхідно додатково встановити пакети aircrack-ng та wireshark. На рисунку 1.4 показаний приклад інтерфейсу.

Проект nzyme використовує адаптери Wi-Fi як спостереження для сканування частот на наявність підозрілої активності, включаючи шахрайські точки доступу і відомі платформи для атак WiFi. Кожен записаний бездротовий кадр аналізується і за необхідності відправляється до системи управління журналами Graylog для довгострокового зберігання.

Рисунок 1.4– Інтерфейс Waidps

Це дозволяє проводити криміналістичне розслідування та реагувати на інциденти. Ви коли-небудь замислювалися, що б ви зробили, якби спіймали зловмисника? За допомогою nzyme ви можете відновити те, що сталося, хто був метою та хто був успішно скомпрометований [11].

Деякі типи попереджень генеруються автоматично. Методи, що використовуються, варіюються від аналізу ймовірної мережної інфраструктури на основі сигнатур до використання відбитків пальців для оцінки ландшафту загроз і встановлення пасток-пасток.

Спочатку потрібно встановити пакет deb або використовувати файл jar і налаштувати вашу систему для роботи. Нам також потрібно налаштувати конфігураційний файл для підключення до Graylog. Graylog використовується для відображення (його можна використовувати як віртуальну машину) та може відображати інформацію на візуальному дисплеї. Його інтерфейс показано на рисинку 1.5.

Рисунок 1.5 – Інтерфейс Nzyme з журналом Graylog

Останньою системою, що розглядається, є AirMagnet Enterprise, яка може вирішувати наступні завдання: AirMagnet Enterprise – Система моніторингу безпеки та продуктивності мереж Wi-Fi. Це гнучке масштабне рішення для забезпечення високої продуктивності та безпеки мереж Wi-Fi з можливістю моніторингу мереж мобільних операторів. Цей програмно-апаратний комплекс не залежить від постачальника та легко інтегрується у існуючі інфраструктури WiFi.

AirMagnet Enterprise дозволяє будь-якій організації ефективно відстежувати умови радіохвиль, своєчасно виявляти джерела загроз для безпеки корпоративної мережі та превентивно блокувати їх.

AirMagnet Enterprise дозволяє віддалено аналізувати трафік Wi-Fi на стандарти 802.11a/b/g/n/ac, активно опитувати всі існуючі канали Wi-Fi, перевіряти доступність точки доступу, можна виконувати спектральний аналіз визначення джерела. Це рішення перешкод, що впливають на продуктивність мережі, дозволяє службам інформаційної безпеки забезпечувати 100% захист 24/7, а відділам інформаційних технологій забезпечуватиме високу продуктивність бездротових мереж.Основні характеристики AirMagnet Enterprise:

– Робота на стандарті 802.11ac;
– Спеціальний метод виявлення вторгнень на базі сигнатур;

– Присутність аналізатора частот, який має змогу знаходити перекриті перекриття канали 802.11 і виявляти радіоперешкоди ;
– звіти про відповідність стандартам HIPAA, PCI DSS, GLBA, DoD, ISO 27001, BASEL 2 і CAD3;

Система складається з сенсорів, сервера і консолі управління зображених на (рис. 1.6).

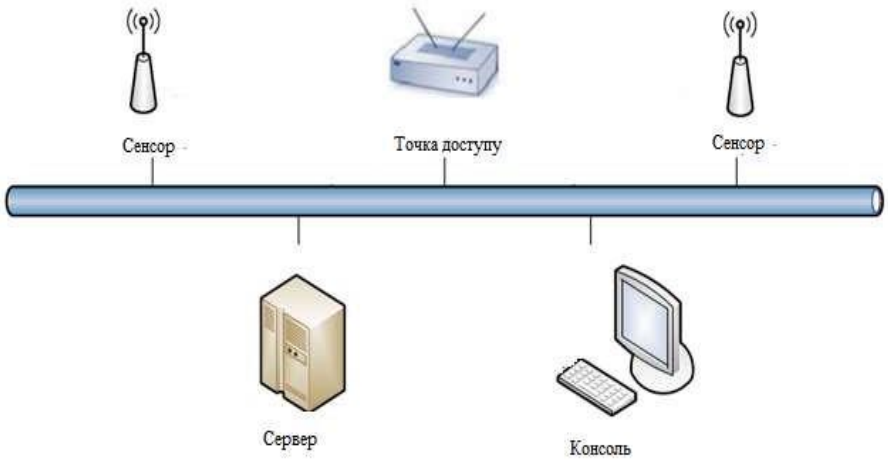


Рисунок 1.6 – Приклад архітектури WIDS

Можливості тріангуляції дозволяють виявити бездротовий пристрій зловмисника. Для цього необхідно імпортувати в систему план поверху та вказати на плані розміщення датчиків та точок доступу. Якщо пристрій виявлено як мінімум трьома датчиками, воно забезпечує приблизне розташування пристрою, але великий обсяг трафіку, що генерується пристроєм, допомагає підвищити точність позиціонування. Крім того, функція тріангуляції може використовуватися для заборони підключення до корпоративних мереж, ініційованих з-за меж периметра захищеного периметра організації [12].Система дозволяє виявити такі бездротові загрози:

– застосування утиліт зламу;

- атака з повтором перехопленого зашифрованого пакета для прискорення розкриття шифрування;
- атаки по словнику на протокол EAP (велика кількість невдалих спроб встановити сесію);
- помилкові точки доступу, створені за допомогою утиліт і маскуються під корпоративні;
- підміна MAC-адреси з метою обходу фільтрів на основі MAC-адрес;
- спотворені кадри стандарту 802.11;
- пряма передача пакетів між клієнтами, що є порушенням політики;
- атака «людина посередині»;

Система має гнучкий функціонал побудови звітів, які можуть складатися за різними стандартами. Недоліком системи є складність її налаштування при потребі широкого функціоналу порівняння засобів для пошуку атак на (табл 1.1).

Для порівняння використана шкала балів 0-7

0 – параметр не задовольняє мінімальним вимогам користування

2 – параметр задовольняє мінімальні вимоги для користування

10 – найкращий рівень параметру серед представлених вище програм

Таблиця 1.1 – Порівняння засобів для пошуку атак

Комплекси/критерії	Аналіз трафіку	Відподвід-ність стандартам	Ціна	Захист	Простота роботи	Сумарна балів (0-50)
AirMagnet Enterprise	7	8	5	8	5	33
Waidps	4	0	6	4	7	21
Nzyme	3	0	7	3	8	22
Avast Free	2	0	8	2	8	20

На жаль, зараз в Україні немає конкретної відповідальності за злом або використання чужих бездротових каналів у рамках закону. Таким чином, користувачі домашніх бездротових мереж можуть бути захищені, адже зловмисник, який отримав несанкціонований доступ до вашої мережі, ще не є злочинцем за законом, і повинен спочатку довести всі правопорушення, знайшовши порушника.

Всі надійні та відомі рішення цієї проблеми не задовольняють потреби звичайних користувачів, які не можуть розгорнути великі та дорогі системи моніторингу безпеки, але при цьому оснащені набором функцій, що дозволяють більш-менш серйозно відстежувати. атака. Також є досить серйозні проблеми з наявністю персоналу, здатного працювати з обладнанням, сумісним із цими комплексами. Це пов'язано з тим, що розгортання системи захисту від атак із використанням вже розгорнутих бездротових мереж може бути надто дорогим та складним [13].

Пройде щонайменше кілька років, перш ніж WPA3, Easy Connect та Enhanced Open стануть стандартами. WPA3 набуде широкого поширення після заміни або модернізації маршрутизаторів. І хоча проблема захисту інформації про вразливості все ще існує, немає жодної гарантії, що WPA3 не містить серйозних вразливостей.

1.4 Постановка задачі

У сучасному світі системи захисту спеціальних завдань не є чимось новим, хоча кожна реалізація спрямована на конкретне завдання, вони страждають на ряд недоліків, що належать до галузі захисту інформації та стійкості до кібератаків. Провівши аналіз існуючих систем захисту під конкретні завдання та порівнявши їх із існуючими аналогами, необхідно розробити унікальну систему захисту на основі веб-ресурсів запису про домашнє насильство, перелік завдань визначено:

1) Розробити модель бази даних – при побудові моделі бази даних було використано загальні принципи побудови баз даних.
2) Розробити метод розпізнавання кіберзагроз в системі збирання та аналізування даних.
3) Розробити комбінований метод захисту веб-застосунків.
4) Провести тестування роботи системи.

У даному розділі було проаналізовано сучасний стан розвитку систем захисту, виявлено деякі основні потенційні уразливості таких систем, а також розглянуто перелік національних стандартів інформаційної безпеки.

Він визначає і розглядає аналоги систем, що захищаються, описує їх сильні і слабкі сторони, проводить порівняльний аналіз і визначає актуальність розробки унікальних систем для конкретних завдань.

Також складено перелік завдань з урахуванням усіх недоліків існуючих аналогів, які необхідно виконати для створення власної захищеної системи збирання та аналізу даних.

2 МОДЕЛЬ ЗАХИСТУ БАЗИ ДАНИХ ОБ'ЄКТІВ ЧУТЛИИХ ДАНИХ

2.1 Розробка захищеної моделі бази даних об'єктів

База даних «Кіберзагрози об'єктів захисту інфраструктури» призначена для використання в автоматизованих системах розпізнавання несанкціонованих впливів на режим роботи об'єктів захисту. Кластерна структура бази даних використовується для зберігання та надання параметрів елементів та об'єктів, для захисту інформації розрахункових режимних параметрів та інформації, що отримується в режимі реального часу від SCADA (існуюча система управління та збору інформації) та EMS (система управління енергоспоживанням). . призначений захисту При побудові основний моделі бази даних використовувалися загальні принципи побудови бази даних [8].

Основний функціональний режим бази даних.

1) Доступність даних. Надає авторизованим користувачам можливість вставляти, редагувати, видаляти та вилучати дані з бази даних.

2) Мета опис даних. Система управління базою даних (далі СУБД) повинна надавати системний каталог, що містить:

- опис даних, які зберігаються в БД;
- опис зв'язків між даними;
- обмеження цілісності даних;
- реєстраційні дані користувачів;
- інша службова інформація.

Завдяки метаданим зовнішні програми можуть отримати доступ до баз даних, спрощуючи розуміння семантики даних, підвищуючи заходи безпеки та виконуючи попередні вимоги для аудиту інформації.

3) Паралельний контроль. Реалізація механізму одночасного розрахованого на багато користувачів (паралельного) доступу до оброблюваних даних з гарантованим коректним оновленням.

Обробка даних усередині транзакції. База даних завжди повинна знаходитись у критичному стані, незалежно від збоїв під час операцій оновлення даних. З цієї причини маніпуляції з даними (в основному вставки, зміни та видалення) згруповані в єдиний блок, який називається транзакцією. Усі оператори транзакцій мають виконуватися коректно та повністю. Тільки в цьому випадку зміни фіксуються у базі даних. В іншому випадку транзакція автоматично відкочується. Тобто стан бази даних відновлюється до моменту, що передує виклик транзакції.

Забезпечення цілісності даних. Усі дані, що містяться в базі даних, мають бути точними та нечасними. Це означає, що дані у таблиці можуть бути змінені лише за затвердженими правилами.В загальному випадку мається на увазі три правила підтримки цілісності даних:

- цілісність доменів;
- цілісність відношень;
- цілісність зв’язків між відношеннями.

Відновлення даних. У разі непередбаченої помилки чи збою, які ушкоджують чи спотворюють дані, СУБД повинна мати можливість відновити пошкоджені дані. По-перше, цю можливість реалізовано з допомогою процедури резервного копіювання.

Обмін даними. СУБД має підтримувати новітні технології та надавати доступ до БД віддаленим ПК.

Контроль доступу даних. Доступ до даних можуть отримати лише зареєстровані користувачі відповідно до привілеїв, призначених адміністратором СУБД.

Узагальнена структурна схема моделі баз даних. Щоб СУБД могла надавати вищезазначені послуги, вона має складатися з набору компонентів, показаних на (рис. 2.1).

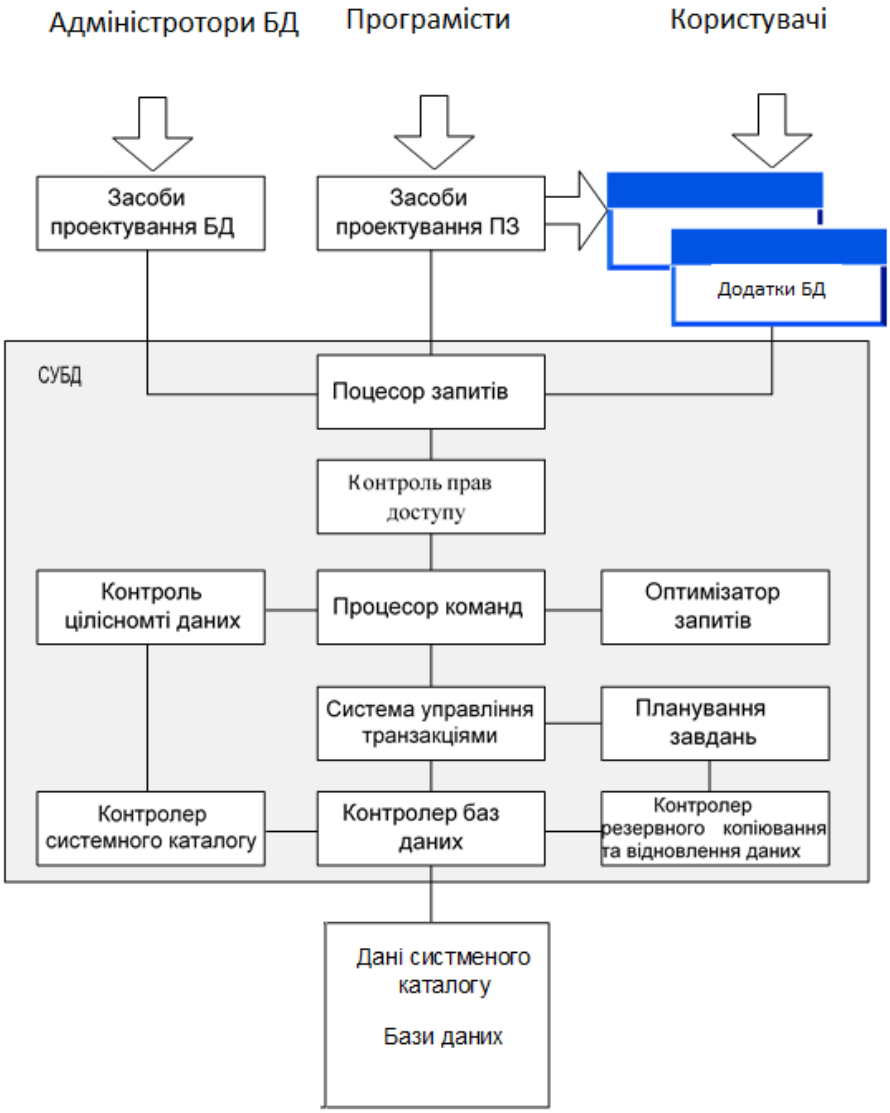


Рисунок 2.1 – Узагальнена структура моделі бази даних

Над вищим рівнем моделі бази даних розташовані споживачі послуг БД:

- адміністратор БД;
- програмісти ПЗ;
- користувачі.

Адміністратори бази даних несуть відповідальність за планування та фактичну реалізацію проектів. Створюйте ключові об'єкти бази даних, визначайте правила підтримки цілісності та неузгодженості даних, керуйте політиками безпеки, аналізуйте операційні процеси проекту та відстежуйте ефективність системи.

Програмісти фактично реалізують концепції баз даних, розробляючи клієнтські програми та звіти. Основними інструментами програмування програм є різні середовища проектування та розробки як мінімум 4-го покоління.

Користувачами послуг баз даних є як фахівці з кібербезпеки, так і персонал об'єктів критичної інформаційної інфраструктури.

Засобом взаємодії між користувачем та базою даних є мова структурованих запитів SQL. Інструменти проектування БД, програмне забезпечення та клієнтські програми надсилають інструкції SQL до СУБД. Ці команди надсилаються обробнику запитів і транслюються до серії низькорівневих команд, доступних ядру СУБД.

У СУБД є модулі, що управляють правами доступу для користувачів із різними правами доступу. Потенційні паролі та логіни клієнтів бази даних, отримані під час реєстрації, звіряються з обліковими записами, що зберігаються у системних каталогах.

Після успішних кроків автентифікації та авторизації користувача модуль керування доступом надає доступ до командного процесора. По-перше, командний процесор гарантує, що вхідні команди не порушують обмеження цілісності даних. За це відповідає модуль контролю цілісності даних.

Контролер системного каталогу задіяний, коли команди перевіряються на відповідність обмеженням цілісності домену, об'єкта та відносини. Збирає метадані, що містять технічний опис бази даних.

Після підтвердження відсутності загроз цілісності процесор передає команду оптимізатор запитів. Завдання оптимізатора – знайти найбільш ефективний спосіб виконання команд, що надходять.

Оптимізовані команди компілюються та передаються до системи управління транзакціями. Система управління транзакціями відповідає за повне і коректне виконання командних блоків, сумісна з планувальниками завдань і забезпечує паралельну розрахунку на багато користувачів даних.

Потім командний блок передається контролеру бази даних. Завданням модуля є організація взаємодії СУБД із файлами БД та файлами системних каталогів. У той самий час він використовує засоби операційної системи до виконання стандартних операцій ввода-вывода.Системний каталог служить для зберігання наступної інформації:

- опис типів даних, що підтримуються БД;
- опис розгорнутих БД (схеми даних) та об’єктів (домени, таблиці, представлення тощо), що входять до них;
- відомості про обмеження цілісності;
- імена та права користувачів, що мають доступ до даних;
- статистичні дані.

Системний каталог реалізований у вигляді окремої бази даних із системними таблицями за умовчанням, прихованими від звичайних користувачів. Доступ до баз даних.

Дозволяється створювати БД (здійснюється за допомогою мови визначення даних DDL).

Дозволяється додавання, оновлення, видалення та читання інформації з БД (за допомогою мови маніпулювання даними DML, яку часто називають мовою запитів).

Модель бази даних передбачає архітектуру доступу клієнт-сервер. Модель клієнт-сервер бази даних передбачає, що база даних розташована на окремому сервері, на якому також встановлена СУБД. На клієнтську станцію встановлюється користувальне програмне забезпечення та налаштовується мережевий доступ до сервера баз даних. Клієнтські комп'ютери надсилають на сервер запити, написані мовою SQL. Сервер отримує оператор SQL, обробляє

його, а потім повертає результати виконання на клієнтський комп'ютер. Узагальнена архітектура «клієнт-сервер» наведена на рисунку 2.2.

Рисунок. 2.2 – Узагальнена архітектура «клієнт-сервер»

Функціональні переваги клієнт-серверної архітектури моделі бази даних:

1) Підвищена доступність бази даних. Оскільки сервер є відкритою системою, на клієнтських комп'ютерах можуть працювати різні операційні системи та інше програмне забезпечення.

2) Виділені сервери СУБД можуть забезпечувати паралельну розраховану на багато користувачів обробку даних.

3) Основні правила збереження цілісності даних та невзаємності прописані одному сервері СУБД. Ці правила застосовуються до всіх клієнтських робочих станцій.

4) Гнучкий підхід до використання ресурсів комп'ютерної мережі;

5) Єдині правила безпеки всім учасників інформаційного взаємодії.

6)Наявність основного стандарту мови взаємодії SQL відкрило можливості програмного забезпечення різних виробників звертатися до серверів баз даних.

7) Єдині правила ведення та управління базами даних.

Адміністратори даних виконують організаційні функції, а також несуть відповідальність за управління даними (планування бази даних, розробка стандартів та правил) та концептуальне проектування бази даних.

Основне завдання менеджера даних – допомогти сформувати корпоративну стратегію побудови інформаційної системи (ІВ).

Клієнтський комп'ютер надсилає серверу запит, побудований на основі мови SQL. Отримавши та опрацювавши інструкцію SQL, сервер повертає клієнтському комп'ютеру результати її виконання.

Контроль за впровадженням та виконанням на всіх етапахжиттєвого циклу ІС політик, процедур та стандартів з коректного створення, використання та розповсюдження даних;

- розробка концептуальних моделей даних;
- планування процесу створення бази даних.
- Встановлення вимог до використовуваних даних;
- встановлення правил збору, зберігання та подання даних;
- визначення політики інформаційної безпеки;
- розробка концептуальних та логічних моделей баз даних.
- взаємодія з ADB та програмістами для забезпечення відповідності розроблюваних баз даних та клієнтських додатків існуючим стандартам та вимогам;
- управління ІБ та модернізацією програмного забезпечення;
- забезпечити цілісність необхідної документації;

Взаємодія з користувачами бази даних.

Адміністратори бази даних несуть відповідальність за її реалізацію (включаючи фізичне проектування).

Забезпечення безпеки та цілісності даних, обслуговування системи та забезпечення оптимальної продуктивності СУБД та додатків.

Основні завданя адміністратора який відповіда за базу даних:

- вибір цільової СУБД;
- розробка логічних моделей баз даних;
- забезпечення необхідного рівня захисту та цілісності даних;
- тестування бази даних;
- робота з базою даних;
- навчання користувачів роботі з базами даних;
- системне адміністрування СУБД;
- управління продуктивністю, резервне копіювання;
- відновити;
- дублювання;
- розмежування прав доступу користувачів.
- ведення технічної документації;
- підтримка БД.

Керуючись концептуальною моделлю даних та затвердженими політиками та стандартами, розробники баз даних розробляють логічну модель даних, а потім фізичну модель даних.

Прикладні програмісти здійснюють розробку та впровадження клієнтського ПЗ. До їх завдань входить:

- організація доступу клієнтського додатку до БД;
- проектування інтерфейсу користувача;
- наповнення додатку необхідним функціоналом;
- тестування та відладка додатків;
- супроводження додатків в період їх експлуатації.

Користувач є особою, що користується послугами БД, зокрема використовує інформації, що міститься в БД, у своїй повсякденній службовій діяльності.

Нижче наведена структура бази даних кіберзагроз об’єктів критичної інформаційної інфраструктури (рис. 2.3)

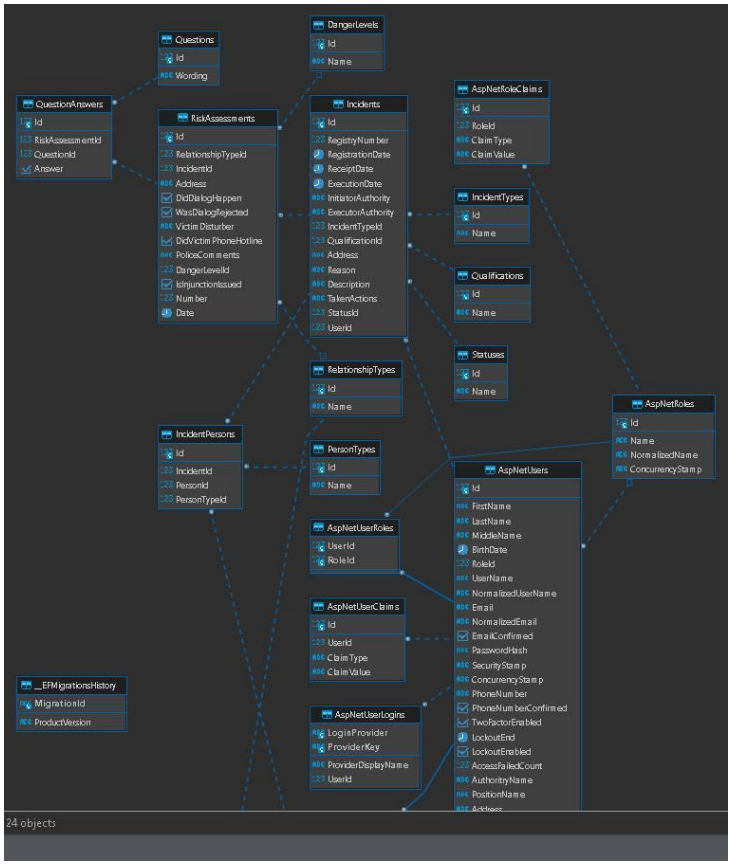


Рис. 2.9 – Загальна структура бази даних

Нижче наведено приклад об'єкта таблиці бази даних наякій заображено усі основні положення та додатку : тип, спецефікація, особливості а також тип захисту (табл. 2.1).

Таблиця 2.1 – Об’єкт таблиці бази даних

Тип	Кіберзагроза
Threat	Denial-of-Service, DoS
Specifications	Кількість одночасних підключень до серверу
Specifications	Кількість пакетів з однаковою адресою відправника та отримувача
Specifications	Швидкість обробки запитів
Specifications	Затримка між запитами від одного користувача
Properties	Доступність
Properties	Спостережність
Protection	Аналіз трафіку
Protection	Блокування шкідливих IP-адрес
Protection	Фільтрація трафіку
Protection	Адекватні апаратні потужності та канали зв’язку

Розробники бази даних, керуючись концептуальною моделлю данихта прийнятими політиками та стандартами, здійснюють розробку логічної, а потім фізичної моделі даних. Фізичні характеристики логічних атрибутів наведені в таблиці 2.2.

Таблиця 2.1 – Фізичні характериститики таблиці

№	Назва атрибуту	Фізичний формат	Характеристи
1.	Threats	TXT	Назва загрози
2.	Properties	TXT	Властивості інформації
3.	Protection	TXT	Заходи протидії
4.	Specificatio ns	TXT	Назва типа документу
5.	Thre_ID	BINARY	Ідентифікатор загрози
6.	Prop_ID	BINARY	Ідентифікатор властивості інформації
7.	Prot_ID	BINARY	Ідентифікатор заходу протидії
8.	Spec_ID	BINARY	Ідентифікатор характеристики загрози

Систематизація кібератак є деревоподібною структурою, відгалуженнями якої є етапи, дії і події, що є будівельними блоками атаки. Показано структуру та зміст кожного з цих понять.

Представлено таксономію кіберзагроз. У його основі лежить комбінований підхід вирішення завдань класифікації. Розроблена таксономія запроваджує ієрархію відносин через деревоподібне розкриття категорій. Як окремий і відокремлений об'єкт вводиться важливе поняття «етапи атаки» для природного опису поширених сьогодні багатоетапних атак.

2.2 Розробка методу розпізнавання кіберзагроз системи збирання та аналізування даних

На основі результатів аналізу наукової літератури [17] та наявного практичного досвіду побудови систем захисту інформації розроблено метод

розпізнавання кіберзагроз інформаційної безпеки комп'ютерних систем та мереж об'єктів.

Цей метод заснований на моніторингу трафіку, що надходить із глобальної мережі Інтернет у відомчі телекомунікаційні системи. Цей метод призначений для використання в багаторівневих системах виявлення передбачуваних ефектів, які відстежують, аналізують та обробляють індикатори вхідного трафіку. Цей метод реалізує три рівні аналізу впливу:

– Сканування трафіку в автоматичному порядку та визначення типу і класифікації протоколу мережевої дії.

– Аналіз та виявлення підозрілих факторів, таких як відмова в обслуговуванні, заміна IP-адреси, вразливості (слабкі місця) протоколу мережевої взаємодії, вразливості додатків (слабкі місця).

– Атаки на паролі (спроби підбору), захоплення привілеїв (диверсія), спроби впровадження шкідливих програм типу «троянських коней», аудити мережі (моніторинг), приховані дії.

У методах підписів системні події представлені таким чином: у вигляді рядка літер алфавіту. Суть цих методів полягає у створенні серії сигнатур атак у формі регулярних атак правила, засновані на виразах (регулярних виразах) або відповідності.

Перевірка відповідності шаблону та спостережуваної відповідності події до цих виразів. типові представники системи, Snort і Suricata [12] реалізують такі методи.Головною перевагою схеми підпису є:

Виявлення відомих моделей аномальних подій має виконуватися якомога ефективніше. У той же час, однак, використання великої бази даних сигнатур негативно впливає на продуктивність системи виявлення [1].

Якщо будь-яка частина коду програми, що відображається, збігається з відомим кодом (сигнатурою) вірусу в її базі даних, антивірусна програма виконує одну з наступних дій:

1) Знищити заражений файл.

2) Перемістити файл до карантину — запобігає виконання файлу, щоб запобігти подальшому поширенню вірусу.
3) Спробував відновити файл і видалив вірус з тіла файлу.

Цей метод має кілька переваг.

– Перевірена надійність. Цей метод використовується давно, що дозволило розробити основні механізми та засоби ефективного виявлення та знешкодження найбільш поширених вірусних програм.

Сучасні обчислювальні ресурси дозволяють точно та оперативно порівнювати програмний код із сигнатурами антивірусних баз.

– Сигнатурна проблема лавинного зростання. Причиною цього явища є збільшення кількості нових вірусів та зміна їх характеристик. В результаті база сигнатур розростається до шалених розмірів, нівелюючи другу перевагу цього методу: швидкість. Як вирішення цієї проблеми ми розглядаємо застосування спеціальних оптимізацій, коли одна сигнатура описує безліч вірусів. Однак це зменшує першу перевагу.

– Проблеми виявлення нових вірусів. Вважається, що користувачі мало сприяють зростанню вірусної бази. При цьому виявлення нових вірусів зазвичай вважається проблемою розробників антивірусів. Однак такий підхід порушує принцип "Безпека стосується всіх".

У той самий час, як було зазначено вище, збільшення антивірусної бази неминуче призводить до уповільнення швидкості обробки інформації.

Евристичний аналіз – метод виявлення шкідливих програм, антивірусні програми відстежують усі дії, що виконуються програмою, що тестується. У ході евристичного аналізу відстежуються потенційно небезпечні дії, типові для другого типу вірусів та шкідливих програм.

Евристичні аналізатори в сучасних антивірусних програмах можуть виявляти нові невідомі віруси доти, як вони почнуть діяти, контролюючи поведінка програм, що перевіряються.

Проте евристичний аналіз дає повної гарантії виявлення нових вірусів. Крім того, евристичні аналізатори можуть сприйняти «нешкідливі» програми за шкідливі. Це відбувається, коли програма виконує дії, типові для вірусів та інших типів шкідливих програм.

Відомі методи евристичного пошуку шкідливих програм. Суть його полягає у аналізі поведінки всіх запущених виконання програм. При виявленні будь-якої підозрілої поведінки програми в процесі роботи системи, тобто якщо програма починає виконувати дії, що не належать до його функціонального призначення і раніше не відбувалися, спрацьовує сигнал тривоги про небезпеку, а евристичний модуль повідомляє користувачів про потенційні загрози.

До переваг цього можна віднести наступне.

– Цей метод є перспективним напрямом розвитку. Враховуючи сучасні тенденції розвитку комп'ютерних систем та технологій штучного інтелекту, можливості евристичних модулів у майбутньому зростатимуть, що неминуче підвищить рівень безпеки як програмних, так і апаратних засобів обробки інформації.

– На відміну від першого способу, модуль евристики може реагувати на загрози, для яких немає інформації в сигнатурній базі даних.

У той самий час цей метод має деякі недоліки.

– Хибна активація як реакція на безпечні події. З урахуванням людського чинника це може призвести до того, що користувач може вимкнути евристичний модуль після кількох помилкових спрацьовувань, що неминуче знижує рівень захисту інформаційної системи.

– Однією з характеристик евристичного модуля є проблема надмірного споживання обчислювальних потужностей. Іншими словами, антивірусне програмне забезпечення несправедливо займає пам'ять і ресурси процесора, що призводить до погіршення продуктивності всієї інформаційної системи. В результаті відбувається вказана вище подія – користувач вимикає модуль евристики.

Відомі способи моніторингу телекомунікаційних мереж, що ґрунтуються на використанні міжмережевих екранів. Захисні механізми ранніх версій брандмауерів ґрунтувалися на попередньо налаштованих знаннях про додатки, мережеві відносини між додатками та механізми забезпечення існуючих відносин. У цьому випадку обмінюватися даними можуть тільки перевірені хости та програми. У нових версіях брандмауера додано механізм глибокої перевірки пакетів (DPI). Це створює гібрид брандмауера та антивірусу з можливістю перевірки характеристик даних, що проходять через брандмауер.

Багато програм використовують небезпечні методи для підключення до віддалених комп'ютерів або серверів, залишаючи "дірки" та вразливості для входу ззовні.

Суть роботи міжмережевого екрану полягає в тому, щоб контролювати як вхідний, так і вихідний трафік, обмежуючи можливість встановлення з'єднань із зазначеними віддаленими ресурсами. Найпоширеніший спосіб захисту

Білий та чорний список мережевих ресурсів. Чорний список – це список мережних ресурсів, до яких немає доступу. Білий список – це список ресурсів, до яких є лише доступ. Зрозуміло, що метод білого списку безпечніший, але з іншого боку сильно обмежує функціональність користувачів та додатків.

– Налаштування брандмауера забезпечують можливість мережевої взаємодії лише з перевіреними ресурсами та ізолюють усі потенційно небезпечні неперевірені мережеві ресурси.

– Може бути встановлений на мережевий шлюз для локальної мережі, сервер, що надає доступ в Інтернет комп'ютерам, що входять до єдиної локальної мережі організації без витрати обчислювальних ресурсів на машині користувача.

Як і попередній метод, цей метод також має недоліки. Це логічно випливає із його переваг. Персонал, який експлуатує та обслуговує брандмауери, повинен мати високу кваліфікацію та глибокі знання мережевих протоколів та функцій мережевих додатків.

Рівень кваліфікації технічного персоналу важливий під час роботи з міжмережевими екранами, оскільки екрани, які працюють із налаштуваннями «за умовчанням», дуже неэффективны.

В основі цього методу лежить завдання розробки систем моніторингу та інтелектуального виявлення підозрілих впливів на об'єкти мереж інформаційно-комунікаційних систем. Застосовуючи три рівні моніторингу та фільтрації трафіку, можна блокувати ширший спектр загроз, ніж згадані вище існуючі методи запобігання мережевим атакам.

Це завдання вирішується наявністю системи, що виявляє підозрілі ефекти аналізу та фільтрації трафіку на трьох згаданих вище рівнях.

Таким чином, в даному розділі удосконалено підходи до класифікації кіберзагроз, що дало змогу провести більш детальну їх класифікацію, розробити актуальну таксономію кіберзагроз, скласти матрицю кіберзагроз та розробити модель бази даних загроз інформаційним об’єктам захисту комп’ютерних систем.

Розроблена модель бази даних загроз інформаційним об’єктам захисту комп’ютерних систем зарахунок використання параметрів загроз, визначених та класифікованих з використанням розробленої таксономії кіберзагроз, їх характеристик, параметрів заходів протидії, параметрів властивостей інформації, що підлягає захисту, дозволяє розробити базу даних кіберзагроз інформаційній безпеці комп’ютерних мереж та систем об’єктів критичної інфраструктури.

3 МЕТОД ЗНЕШКОДЖЕННЯ ЗАГРОЗ БЕЗПЕКИ ВЕБ-ЗАСТОСУНКІВ

3.1 Розробка комбінованого методу захисту веб-застосунків.

Ненадійне програмне забезпечення підриває безпеку критичних інфраструктур, що стосуються наприклад, охорони здоров'я, оборони, енергетики або фінансів. Програмне забезпечення стає складнішим, пристроїв, підключених до мережі, стає більше, тому важливість забезпечення безпеки програм зростає експоненційно. Швидкий розвиток методів розробки ПЗ призводить до необхідності швидкого та безпомилково виявляння, а також усування загроз , що найчастіше виникають.

Майже будь-яке джерело даних може виявитися вектором для впровадження: змінні оточення, параметри, зовнішні та внутрішні веб-служби, а також усі типи користувачів. Впровадження стають можливими, якщо зловмисник може відправляти інтерпретатору шкідливі дані.

Для запобігання впровадженням необхідно ізолювати дані від команд та запитів:

- Використовуйте безпечний API, що виключає застосування інтерпретатора або надає параметризований інтерфейс, або використовуйте інструменти об'єктно-реляційного відображення (ORM).

Примітка: навіть параметризовані процедури, що зберігаються, можуть призвести до SQL-впровадженням, якщо PL/SQL або T-SQL дозволяють приєднувати запити та дані або виконувати шкідливий код з EXECUTE IMMEDIATE або exec().

- Реалізуйте на сервері білі списки для перевірки вхідних даних. Це, звичайно, не забезпечить повного захисту, оскільки багато додатків використовують спецсимволи, наприклад, в текстових областях або API для мобільних програм.

– Для решти динамічних запитів реалізуйте екранування спецсимволів, використовуючи відповідний інтерпретатору синтаксису.

Елементи структури SQL, такі як назви таблиць або стовпців, не можна екранувати, тому надані користувачами назви становлять небезпеку. Це звичайна проблема програм для складання звітів. Використовуйте у запитах LIMIT або інші елементи керування SQL для запобігання витоку даних.

Приклад підготовленого оператора захисту C# NET. Створення та виконання запиту не змінюється. Для запобігання потрібно зробити передачу параметрів в запит за допомогою виклику Parameters.Add().

```
String query = "SELECT account_balance FROM user_data WHERE user_n
ame = ?";
try {
  OleDbCommand command = new OleDbCommand(query, connection);
  command.Parameters.Add(new OleDbParameter("customerName",
CustomerName Name.Text));
  OleDbDataReader reader = command.ExecuteReader();
  // …
} catch (OleDbException se) {
  // error handling
}
```

Зловмисники мають доступ до сотень тисяч дійсних комбінацій імен та паролів для атак на облікові записи, списків стандартних облікових даних адміністраторів, інструментів для автоматизації атак методом підбору та атак за словниками. Атаки на сесії добре вивчені, особливо в частині діючих токенів сесій.

Реалізація багатофакторної аутентифікації для запобігання автоматизованим атакам, атакам на облікові записи та методом підбору, а також повторного використання вкрадених облікових даних. Основні методи захисту:

– Використання унікальних облікових записів які створюються за замовчуванням.

- Реалізація перевірки надійності паролів, наприклад.
- Встановлення довжини, складність та періодичність зміни паролів у відповідно до керівництва NIST 800-63 B або будь-якою іншою сучасною парольною політикою.
- Забезпечення захисту реєстрації, відновлення облікових даних та API від атак методом перерахування, використовуючи у всіх відповідях однакові повідомлення.
- Обмеження або збільшення інтервалів між невдалими спробами входу. Реєструйте всі невдалі спроби та повідомляйте адміністраторів при виявленні атак на облікові дані, методом підбору чи будь-яких інших атак.
- Використовування серверих, надійних вбудованих менеджерів сесій, генеруючі після входу в систему нові, випадкові ідентифікатори високим ступенем ентропії. Ідентифікатори сесій не повинні бути в URL, а повинні безпечно зберігатися і анулюватися після виходу з системи, простою або настання абсолютного тайм-ауту.

Щоб дозволити користувачеві надіслати запит на скидання пароля, вам знадобиться певний спосіб ідентифікації користувача або засіб зв’язку з ним через бічний канал.

Це можна зробити будь-яким із наведених нижче методів:

- URL-токени .
- PIN-коди
- Офлайн методи
- Запитання безпеки .

Ці методи можна використовувати разом, щоб забезпечити більший ступінь впевненості, що користувач є тим, за кого себе видає. Незважаючи ні на що, ви повинні переконатися, що користувач завжди має спосіб відновити свій обліковий запис, навіть якщо для цього потрібно звернутися до служби підтримки та підтвердити свою особу співробітникам.

Зловмисники можуть експлуатувати вразливі обробники XML через завантаження XML або впровадження шкідливого контенту в XML-документи, використовуючи вразливий код, залежності чи компоненти, для запобігання XXE необхідно:

- використовувати, по можливості, простіші формати даних, наприклад, JSON, та уникати серіалізації критичних даних;
- встановити виправлення чи оновлення для всіх бібліотек та обробників XML, що використовуються програмою або ОС. Використати перевірки залежностей. оновити SOAP до версії 1.2 або вище;
- вимкнути обробку зовнішніх сутностей XML і DTD у всіх XMLобробниках програми, згідно з
- реалізувати на сервері (за білими списками) перевірку, фільтрацію або очищення (екранування) вхідних даних для запобігання влученню шкідливих даних у XML-документи, заголовки чи вузли;
- переконайтеся, що функція завантаження XML або XSL перевіряє вхідні файли з використанням XSD або іншої подібної методики;
- аналізувати код масштабних та складних додатків збезліччю вбудовуваних компонентів вручну, хоча інструменти SAST можуть допомогти виявити XXE у вихідному коді.

Якщо виконання цих вимог неможливо, спробуйте використовувати віртуальні патчі, шлюзи безпеки API або файрволи веб-додатків (WAF) для виявлення, моніторингу та блокування XXE-атак.

Експлуатація контролю доступу є основною навичкою зловмисників. Інструменти SAST та DAST можуть виявити відсутність контролю доступу, але не можуть перевірити його працездатність за його наявності. Наявність контролю доступу можна виявити вручну, а його відсутність можна виявити автоматично у деяких фреймворках.

Контроль доступу ефективний лише при реалізації через перевірений код на стороні сервера або безсерверний API, де атакуючий не може змінювати перевірки прав доступу або метадані. Для того щоб запобігти вище вказаним проблемам було впровадженно такі пункти:

- забороняти доступ за промовчанням, за винятком відкритих ресурсів;
- реалізувати механізми контролю доступу та використовувати їх у всіх додатках, а також мінімізувати міждоменне використання ресурсів;
- контролювати доступ до моделей, використовуючи володіння записами, а не можливість користувачів створювати, переглядати, оновлювати або видаляти будь-які записи;
- використовувати моделі доменів для реалізації спеціальних обмежень, які стосуються додатків;
- вимкнути виведення списку каталогів веб-сервера, а також забезпечити відсутність метаданих файлів (наприклад, .git) та файлів резервних копій у кореневих веб-каталогах;
- реєструвати збої контролю доступу та повідомляти адміністраторів при необхідності (наприклад, якщо збої повторюються); обмежувати частоту доступу до API та контролерів для мінімізації шкоди від інструментів автоматизації атак; анулювати токени JWT на сервері після виходу із системи.

Розробники та інженери з контролю якості ПЗ повинні проводити функціональну перевірку контролю доступу та тестувати інтеграцію.

Перевіряйте дозволи на кожен запит**.**

Дозволи мають бути правильно перевірені для кожного запиту, незалежно від того, чи був запит ініційований сценарієм AJAX, сервером чи будь-яким іншим джерелом. Технологія, яка використовується для виконання таких перевірок, повинна передбачати глобальну конфігурацію всієї програми, а не

застосовувати її окремо до кожного методу чи класу. Пам’ятайте, що зловмиснику потрібно знайти лише один шлях. Навіть якщо «пропущено» лише одну перевірку контролю доступу, конфіденційність та/або цілісність ресурсу може бути під загрозою. Правильної перевірки дозволів лише для більшості запитів недостатньо. Конкретні технології, які можуть допомогти розробникам виконувати такі послідовні перевірки дозволів, включають наступне:

- Java/Jakarta EE Filters , включаючи реалізації у Spring Security
- Проміжне ПЗ у Django Framework
- Фільтри .NET Core
- Проміжне програмне забезпечення в Laravel PHP Framework

Автоматизовані інструменти можуть виявляти та експлуатувати всі три види міжсайтового виконання сценаріїв, фреймворки для їх експлуатації можна знайти у відкритому доступі.

Необхідно реалізувати процес безпечного встановлення, включаючи:

- відтворюваність процесів для швидкого створення безпечних, ізольованих середовищ. Засоби для розробки, контролю якості та експлуатації мають бути налаштовані однаково, але мати різні облікові дані. Процеси повинні бути автоматизовані для мінімізації витрат за створення нових безпечних середовищ;
- використання платформ лише з необхідним набором функцій, компонентів, документації та зразків. Видаліть чи не встановлюйте зайві компоненти чи фреймворки;
- перевірку та актуалізацію параметрів налаштування безпеки в відповідно до бюлетенів, оновлень і виправленнями (див. A9:2017-Використання компонентів з відомими уразливостями), а також перевірку дозволів хмарних сховищ (наприклад, для контейнерів S3);

– створення сегментованої архітектури програми, що забезпечує ефективне розмежування компонентів або клієнтів за допомогою контейнеризації або хмарних груп безпеки;
– використання безпечних директив для клієнтів, наприклад, безпечні заголовки;

Наступні фрагменти HTML демонструють, як безпечно відтворювати ненадійні дані в різних контекстах таблиця 3.1.

Таблиця 3.1 – Фагменти HTML для відтворення не надійних файлів

Тип даних	Контекст	Зразок коду	Тип захисту
String	HTML Body	UNTRUSTED DATA	Кодування об’єктів HTML (правило №1).
String	Safe HTML Attributes	<input type="text" name="fname" value="UNTRUSTED DATA ">	Агресивне кодування об’єктів HTML (правило №2), розміщуйте ненадійні дані лише в списку безпечних атрибутів (перерахованих нижче), суворо перевіряйте небезпечні атрибути, такі як фон, ідентифікатор та ім’я.
String	GET Parameter	<ahref="/site/search?value=UNTRUSTED DATA ">clickme	Кодування URL (правило №5).
String	Untrusted URL in a SRC or HREF attribute	a href="UNTRUSTED URL ">clickme <iframe src="UNTRUSTED URL " />	Канонічне введення, перевірка URL-адреси, перевірка безпечної URL-адреси, лише URL-адреси http і HTTPS із білого списку (уникайте використання протоколу JavaScript для відкриття нового вікна), кодувальник атрибутів.
String	CSS Value	HTML Selection	Сувора перевірка структури (правило №4), шістнадцяткове кодування CSS, хороший дизайн функцій CSS.
HTML	HTML Body	UNTRUSTED HTML	Перевірка HTML (JSoup, AntiSamy, HTML Sanitizer...).
String	DOM XSS	<script>document.write("UNTRUSTED INPUT: " + document.location.hash);<script/>	Запобігання XSS на основі DOM

Зловмисники часто намагаються експлуатувати невиправлені вразливості, налаштовані по за замовчуванням облікові записи, сторінки, що не використовуються, незахищені файли та каталоги для отримання несанкціонованого доступу чи інформації про систему.

Необхідно реалізувати процес безпечного встановлення, включаючи:

- відтворюваність процесів для швидкого створення безпечних, ізольованих середовищ. Засоби для розробки, контролю якості та експлуатації мають бути налаштовані однаково, але мати різні облікові дані. Процеси повинні бути автоматизовані для мінімізації витрат за створення нових безпечних середовищ;
- використання платформ лише з необхідним набором функцій, компонентів, документації та зразків. Видаліть чи не встановлюйте зайві компоненти чи фреймворки;
- перевірку та актуалізацію параметрів налаштування безпеки в відповідно до бюлетенів, оновлень і виправленнями. Використання компонентів з відомими уразливостями), а також перевірку дозволів хмарних сховищ (наприклад, для контейнерів S3);
- створення сегментованої архітектури програми, що забезпечує ефективне розмежування компонентів або клієнтів за допомогою контейнеризації або хмарних груп безпеки;
- використання безпечних директив для клієнтів, наприклад, безпечні заголовки;
- автоматизацію перевірки ефективності використовуваних конфігурацій та налаштувань у всіх середовищах.

Незважаючи на простоту пошуку вже готових експлойтів для більшості відомих уразливостей, деякі з них вимагають створення спеціальних засобів для їхньої експлуатації.

Необхідно реалізувати процес управління оновленнями:

- видаліть невикористані залежності, а також зайві функції, компоненти, файли та відомості з документації;
- регулярно перевіряйте актуальність версій клієнтських та серверних компонентів (наприклад, фреймворків та бібліотек), а також їх залежностей, використовуючи такі інструменти як versions, DependencyCheck, retire.js і т. п. Слідкуйте за новинами про уразливості на відповідних ресурсах, таких як CVE та NVD.

Використовуйте інструменти аналізу складу програмного забезпечення для автоматизації процесу. Підпишіться на розсилки про вразливості, що стосуються використовуваних вами компонентів, а також:

- завантажуйте компоненти з офіційних джерел безпечних посиланням. Віддавайте перевагу підписаним пакетам для зниження ризику встановлення зміненого чи шкідливого компонента;
- стежте за бібліотеками та компонентами, які не підтримуються або не отримують оновлення безпеки. Якщо оновлення не можливе, спробуйте використовувати віртуальні патчі для виявлення або запобігання експлуатації відомих уразливостей.

Кожна організація має забезпечити відстеження, пріоритизацію та застосування оновлень або змін у конфігурації на протягом усього життєвого циклу програми чи лінійки додатків.

Експлуатація недоліків журналування та моніторинг лежить в основі майже всіх великих зломів. Під час проведення атак зловмисники покладаються на відсутність контролю та своєчасне реагування на інциденти.

Виходячи з критичності даних, що зберігаються або оброблюються додатком, необхідно:

- реєструвати всі помилки входу, доступу та перевірки даних настороні сервера із зазначенням контексту, достатнього длявиявлення підозрілих чи шкідливих дій, а також зберігати їх для подальшого аналізу;

- реєструвати події у форматі, що найбільш підходить для обробки централізованою службою журналування;
- використовувати контроль цілісності журналів аудиту важливих транзакцій для запобігання підміні або видалення даних, наприклад, за допомогою доступних тільки для додавання таблиць БД; використовувати ефективні системи моніторингу та попередження для своєчасного виявлення підозрілих дій та реагування на них;
- розробити або затвердити посібник з реагування на інциденти та усунення їх наслідків, таке як NIST 800-61 rev2.

При роботі з великим набором програм завдання може здатися нездійсненним. Для вирішення цих проблем було розроблено узагальнений алгоритм протидії атакам рисунок 3.1.

Рисунок 3.1 – Узагальнений алгоритм захисту від загроз безпеці веб-додатків

Щоб створити безпечну веб-програму, необхідно спочатку розробити вимоги до його безпеки. Для цього рекомендується використовувати стандарт підтвердження безпеки програм OWASP (ASVS). При аутсорсингу використовуйте додаток безпеки до контракту на розробку програмного забезпечення від OWASP. Додаток застосовно до договірного права.

Враховуючи всі пункти, викладені в даному розділі, і чітко дотримуючись рекомендацій та послідовність дій, завдяки методу захисту від загроз безпеці веб-додактів можливо захистити веб-застосунок який було розроблено на вимогу головного управління кіберполіції від широко поширених сьогодні хакерів і зловмисників. Звичайно, цей метод буде продовжувати вдосконалюватися й оновлюватися перед появою нових вразливостей і переглядів.

4 РОЗРОБКА ПРОГРАМНОГО ДОДАТКУ ТА ЕКСПЕРЕМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ

4.1 Обґрунтування вибору засобів реалізації програмного засобу

Всі плюси та мінуси технології випливають із цього факту. Величезне співтовариство, тисячі бібліотек на всі випадки життя, чудовий туллінг все є. Коли приходиш у дотне, у тебе немає жодних проблем. Є Visual Studio і Rider – надсучасні IDE, які самі вміють створювати всі основні шаблони проектів, завантажувати все необхідне, щоб вони заробили, і збирати їх натисканням кнопочки.

Тулінг по-справжньому дорослий. Лінтери робилися довго і вдумливо, компілятор набитий фічами під зав'язку і покриває навіть неочевидні сценарії складання.

Очевидні речі тут роблять очевидним чином, а на всі популярні проблеми є популярні рішення. Є мільйони запитань та відповідей на стековерфлоу. Якщо ти не можеш зробити щось саме в дотнеті, тебе, швидше за все, забанили в гугле. Дуже легко знайти собі знайомих дотнетчиків, які допомагатимуть зі складними особливостями платформи. Написано тисячі книг.

Процес вивчення налагоджений роками, всі знають, що читати, як робити, у кого запитувати. При цьому сама платформа не є суперскладною. Міцний мідл іншою мовою вже через пару днів зможе писати стерпний код на C#, поки справа не торкнеться багатопоточки, роботи з пам'яттю чи оптимізації.

C# – потужна мова з купою можливостей.

Корна либа досить багата – з коробки ти в дотнеті отримуєш тисячі утиліт, структур даних, і всіляких таких речей. Вирішувати стандартні завдання тут справді легко – все є, про все подумали за тебе. Хороший інтероп із плюсами докидає зверху купу плюсових ліб, обгорнутих у дотнетні — на випадок, якщо тебе змусять молотити байти в якомусь софті, пов'язаному з відео чи звуком.

Мова досить сучасна, всі найпопулярніші фічі на ринку в ньому або є, або плануються. Синтаксис легше ніж у Java, важчий ніж у сучасних мовах на зразок Котліна. Шарповий код при компіляції перетворюється на проміжну мову, яку виконує JIT-компілятор на клієнтській машині — у цьому є свої переваги, але такий принцип роботи не залишає шансів писати настільки ж швидкий код, як на плюсах чи голанзі. Але C# все ще досить швидкий у своїй категорії. Якщо просто сісти і писати нехитрий код, дуже довго не впиратимешся в якісь несподівані проблеми у продуктивності самої мови.

Мова не супер швидка — але є величезний простір для оптимізації. Структури — щоб зберігати дані на стеку, АПІ збирача сміття, щоб оптимізувати його роботу в конкретних кейсах, є unsafe — щоб попрацювати з покажчиками та оптимізувати якийсь ботлнек. Багаті інструменти для паралелізму. Є інтринсики для оптимізації під конкретні процесори. Коротше, якщо морочитися – швидкість роботи можна прокачати на раз.

У C# потужна підтримка ООП – тут будь-який код лежить у класах, повний супорт успадкування, а тепер і множинного успадкування через дефолтну реалізацію інтерфейсів. Купа різних модифікаторів доступу, можливість розділяти відповідальність модулів на рівні збирання. Типізація строга статична – все що можна тайпчекается на етапі компіляції, все що не можна – метаінфа про типи їде в рантайм, і може бути прочекана там. Сам дизайн мови ніби спеціально зроблений так, щоб фігачить архітектуру за Бобом та Фаулером. У C# дуже легко пояснити компілятору, які частини системи роблять це, а якісь те, і хто з них про кого знає.

Набагато менше шарпи підтримують функціональну парадигму. Але й тут є чіткий рух уперед. Нещодавно завезли трохи кастрований патерн-матчинг, є топли і майже є рекорди. Давно є лямбда виразу. Найпоширеніший у дотнеті спосіб роботи з колекціями – LINQ вирази – виконаний повністю у функціональному стилі.

Але основну проблему мови це лише посилює. Для сучасного C# дуже багатослівний. Вони завозять нові функціональні фічі, але величезна кількість необхідних синтаксичних надбудов не дозволить їх використовувати так само витончено, як у якомусь OCaml. Тут що далі, то гірше. Страшний вантаж зворотної сумісності легко розрулюється творцями мови технічно, але завжди виливається в ще одне обтяження синтаксису. Мені трохи соромно, коли бачу код мінімально робочої програми на C# — мова буквально відмовляється звільняти мене від написання очевидного.

4.2 Розробка модуля автентифікації користувача

На бекенді для реалізації автентифікації використовуються Entity Framework та ASP.NET Identity. Це підсистеми розроблені Microsoft для проекції бази даних у об’єкти мови програмування C# та для менеджменту користувачів та їхніх ролей. Наприклад клас для ролі буде виглядати наступним чином:

```
public class Role : IdentityRole<uint>
{
    public virtual ICollection<User> Users { get; set; } = null!;

    public Role() : base() { }
    public Role(string roleName) : base(roleName) { }
}
```

ASP.NET Identity надає функцію автоматичного кодування пароля, тобто при додаванні користувача до бази даних, його пароль одразу кодується 1000 разів за допомогою алгоритму HMAC-SHA256, тому навіть розробник не зможе дізнатися його.

Для захисту розробленої системи використовується JWT, тобто токен на основі JSON, який складається з заголовка (header), корисного навантаження (payload) та підпису або даних шифрування. Заголовок це JSON елемент, що описує до якого типу належить даний токен і які методи шифрування використовувались. В нашому випадку це JWT та HMAC-SHA256 Вміст

складається з елемента JSON який описує твердження, в нашому випадку він буде складатися з id користувача, ПІБ, дати народження, пошти та номеру телефону. Структура підпису визначається стандартом JSON Web Signature.

Щоб згенерувати підпис, заголовок та вміст кодуються в Base64, записуються в один рядок через крапку, а потім цей рядок хешується визначеним методом:

SigningCredentials = new SigningCredentials(key, SecurityAlgorithms.HmacSha256Signature)

Даний токен генерується при успішній спробі користувача залогінитись і виглядає наступним чином:

eyJhbGciOiJSUzI1NiJ9.eyJ1c2VyLXNpZCI6IlMtMS01LTIxLTEwMDkxMDg0MjktMTc4ODM3MzUxNC0xMzA1ODQyODMyLTExNTQiLCJjbG91ZEFkbWluIjpmYWxzZSwiYnJva2VyIGlkIjoiRVVDLUNTLTAxIiwidG9rZW4gdHlwZSI6ImFjY2VzcyB0b2tlbiIsInN1YmplY3QiOiJVcmltZSIsImRvbWFpbiI6ImV1Yy1sYWIubG9jYWwiLCJkb21haW4tdHJ1c3QtdHlwZSI6IlBSU1BULlfRE9NQUlOIiwic2Vzc2lvbi1pZCI6IjU0MzAzNmExLTZiZTctNDU1Ni1iMzMzLWMyM2RjMTljYzA1OCIsInN1YiI6IlVyaW1lIiwiaWF0IjoxNjUzODkxNDY4LCJqdGkiOiI1MzI4ZDM0My1jNzU2LTQ3YzAtYjQ4Ny1iNzk3ZGU2ZTg3ZmMiLCJuYmYiOjE2NTM4OTE0NjgsImV4cCI6MTY1Mzg5MzI2OH0.XJrQszyB9K2kzdzGXhGhSDIg2uF5udzyQPGfyZ79N_WhCc530oqAfq4WFZQUMhVk9I-0esA9iJ7M0mQYJiGEcb5NUQ60LJCzVJkF-

Далі цей токен передається фронтенду, який буде додавати його у HTTP запит у «Authorization» header:

```
Authorization: Bearer <token>
```

При обробці запиту, ендпоінти, які позначені атрибутом «[Authorize]», будуть перевіряти даний токен і у випадку успіху надавати користувачу доступи в залежності від його ролі. Якщо ж спроба була невдала, фронтенду повернеться відповідь із статус кодом 401(Unauthorized) та текстовим повідомленням, яке поясне причину, чому спроба не вийшла

Реєстрація побудована за іншим принципом. По-перше для коректної роботи логіки на бекенді потрібно більше полів. По-друге, вся надана інформація додається в базу даних і використовується для генерації токенів у подальшому:

```
var result = await _userManager.CreateAsync(user,
registryRequest.Password);
await _context.SaveChangesAsync();
```

Ключове слово `await` вказує на виконання методу в асинхронній манері.

Якщо при реєстрації виникає помилка то клієнту у відповідь повертається 400 статус код, який означає Bad Request:

```
var registrationErrorResult = new JsonResult(string.Join(" ",
result.Errors.Select(e => e.Description)))
{
     StatusCode = 400
};
return registrationErrorResult;
```

Якщо ж реєстраціє пройшла успішно, то бекенд генерує токен та повертає його клієнту для подальшого доступу до інших маршрутів:

```
var token = _jwtUtils.GenerateToken(userInfoResponse);
return new JsonResult(new {User = userInfoResponse, Token = token});
```

Таким чином реалізована логіка авторизації та автентифікації на бекенді.

Розробка модуля збирання даних про осіб

Модуль збирання даних про осіб налічує методи для повернення, додавання, оновлення та видалення даних про осіб та їхні відносини. Спочатку розглянемо методи додавання:

```
public async Task<IActionResult> AddPersonAsync([FromBody]
PersonAddRequest personAddRequest)
```

На вхід подається великий обсяг даних: ПІБ, дані про документи, дату народження, чи є діти, домашня адреса, робоче місце та id рівня соціального забезпечення. Далі йдуть перевірки правильності введення ПІБ. Якщо кількість введених слів не рівна трьом, то буде повернутий 400 код:

```
var personNames = personAddRequest.FullName.Split(' ');
if(personNames.Length != 3)
{
          var badRequestResult = new JsonResult("Невірно введено
ПІБ")
          {
               StatusCode = 400
          };
          return badRequestResult;
```

```
}
```

Наступним кроком буде перевірка id рівня соціального забезпечення. Бекенд звертається до бази даних та намагається знайти запис з id, який надався при виклику метода додавання особи:

```
var socialSecurity = await _context.SocialSecurities
                        .FirstOrDefaultAsync(s => s.Id ==
personAddRequest.SocialSecurityId);
```

Якщо такий запис є, то робота продовжується, якщо ні, то клієнту повернеться 404 статус код, який означає Not Found:

```
if(socialSecurity == null)
{
var notFoundResult = new JsonResult("Неможливо знайти даний рівень
соціального забезпечення")
{
StatusCode = 404
};
return notFoundResult;
```

Далі створюється об’єкт особи та його полям присвоюються значення надані при виклику метода:

```
var person = new Person();
_mapper.Map(personAddRequest, person);


person.LastName = personNames[0];
person.FirstName = personNames[1];
person.MiddleName = personNames[2];
```

Потім перевіряється інформація про документи особи, а саме чи введені всі поля(id типу документа, орган, що видав, номер документа, дату видачі та серію документа) та чи є тип документу з наданим id. Якщо хоча б одне поле з вищеописаних не введено, то інформація про документи не буде записана до бази. Також, якщо тип документа не було знайдено, то клієнту повернеться статус код 404:

```
var personDocumentType = await _context.PersonDocumentTypes
                .FirstOrDefaultAsync(s => s.Id ==
personAddRequest.DocumentTypeId);
```

```
if(personDocumentType == null)
{
personAddRequest.SocialSecurityId);
var notFoundResult = new JsonResult("Неможливо знайти даний тип
документу особи")           {
StatusCode = 404
           };
            return notFoundResult;
}
```

Далі створена особа додається до бази даних та клієнту повертається id новоствореної особи:

```
await _context.Persons.AddAsync(person);
await _context.SaveChangesAsync();
return new JsonResult(person.Id);
```

По ідентичній логіці працює і метод редагування особи.

Додавання відносин для особи по id відбувається наступним чином. На вхід методу передається список об’єктів, які мають id типу відносин та id особи. Першим ділом перевіряється існування особи для якої користувач додає ці відносини:

```
var person = await _context.Persons.FirstOrDefaultAsync(p => p.Id
== id);
if(person == null)
{
var notFoundResult = new JsonResult("Дану особу не було знайдено")
{
StatusCode = 404
};
return notFoundResult;
}
```

Далі бекенд циклом пробігається по всьому списку переданих об’єктів. За вже описаним принципом перевіряється існування осіб та типів відносин. Продовжується виконання методу створенням об’єкту відносин та додаванням його до бази даних:

```
var relationshipToAdd = new Relationship
{
FirstPersonId = id,
SecondPerson = relationshipPerson,
RelationshipType = relationshipType
};
await _context.Relationships.AddAsync(relationshipToAdd);
await _context.SaveChangesAsync();
```

І клієнту повертається список об’єктів відносин особи, для якої додавалися нові відносини.

Оновлення відносин відбувається ідентичним чином.

Видалення особи відбувається достатньо просто. Перевіряється наявність особи по її id. Якщо така особа є, то вона видаляється:

```
_context.Persons.Remove(person);
await _context.SaveChangesAsync();
```

Повернення особи відбувається наступним чином. Перевіряється наявність особи по id. Якщо вона є, то клієнту повертається об’єкт особи. Якщо немає, то 404 статус код

Отже, модуль збирання даних про осіб має велику кількість методів різної логіки: створення, перегляду, редагування та видалення осіб.

Даний модуль працює по логіці ідентичній той, яка використовується у попередньому, але з деякими відмінностями. Перше це поля, які подаються на вхід у методи додавання та оновлення інцидентів(номер реєстрації, дата реєстрації, орган ініціатор, орган виконавець, id типу події, id кваліфікації, адреса, причина, опис, вжиті заходи, словник осіб, причетних до цієї події). Друге – додаткова логіка обробки словника осіб, причетних до події. Про неї і буде описано в даному розділі

У словника ключ – це id типу особи у інциденті(1 – жертва, 2 – кривдник), а значення – список id осіб вказаного типу. Коли словник починає оброблятися, то спочатку перевіряється наявність типу особи у інциденті по id. Якщо запису з наданим id не існує, то клієнту повертається статус код 404:

```
if (!_context.PersonTypes.Any(pt => pt.Id == 
addIncidentPersonsTypeId))
{
          var NotFoundResult = new JsonResult("Даний тип особи у 
подіі не           було знайдено")
          {
               StatusCode = 404
          };
          return NotFoundResult;
}
```

Далі перевіряється кількість наданих осіб кожного типу. Якщо у якогось типу немає взагалі осіб, то повертається 400 код:

```
if(addIncidentPersonsCount.Any(c => c == 0))
{
          var BadRequestResult = new JsonResult("Не дано осіб для
якоїсь з           груп")
          {
               StatusCode = 400
          };
          return BadRequestResult;
}
```

Якщо у списку id осіб вказаного типу є 2 або більше однакових id або один і той самий id існує у декількох списках, тобто одна й та сама особа має одночасно декілька типів у інциденті, то також повертається 400 код:

```
if(addIncidentPersonsIds.Count(id => id == addincidentPersonId) >
1)
{
          var BadRequestResult = new JsonResult("Дана особа не може
" +"бути обрана двічі")
          {
               StatusCode = 400
          };
          return BadRequestResult;
}
```

Далі перевіряється наявність у базі даних кваліфікації, типу інциденту по їхніх id. Якщо все добре, то далі створюється об’єкт інциденту, всі отримані значення присвоюються його полям, а також його статус встановлюється як «Необроблено». Він додається в базу даних та повертається клієнту його id:

```
await _context.Incidents.AddAsync(incident);
await _context.SaveChangesAsync();
return new JsonResult(new IncidentAddResponse{ Id = incident.Id });
```

Метод оновлення працює по ідентичному принципу, тільки є одна додаткова перевірка: якщо поле «Вжиті заходи» було заповнено, то інциденту виставляється статус «Виконано»

Перегляд інциденту відбувається по наступній логіці: якщо інцидент з вказаним id існує, то він повертається клієнту і його статус стає «Виконується». Якщо ж такого інциденту не існує, то клієнту повертається статус код 404.

Отже, модуль збирання даних про інциденти має схожу до попереднього модуля архітектуру, але є відмінності у реалізації методів перегляду, створення, редагування інцидентів.

4.2 Тестування роботи додатку частини

Тестування розпочинається з маршруту **POST «users/register»**. Спочатку протестуємо випадок, коли все введено правильно:

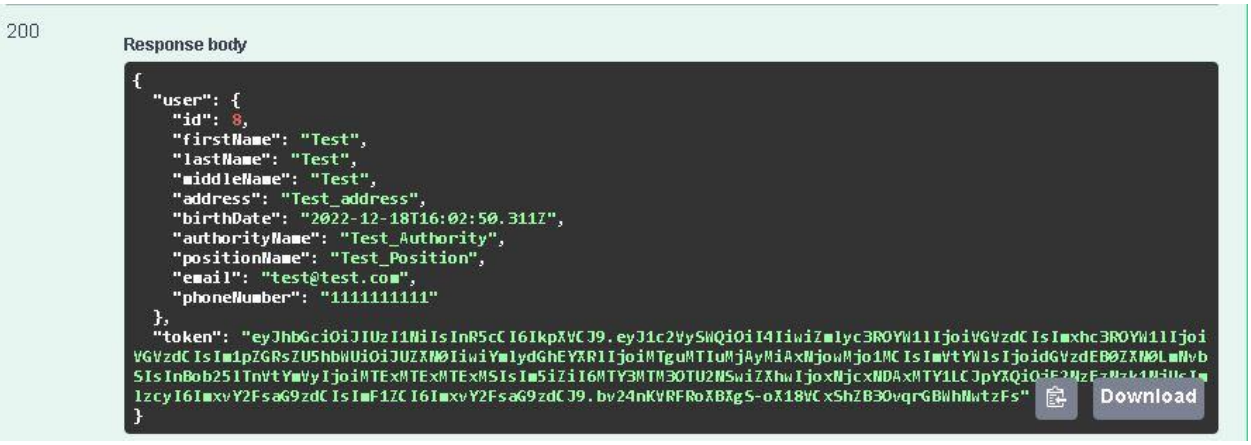


Рисунок 4.1 – Відповідь маршруту POST «users/register» при правильно введених даних

Також протестуємо випадки, коли деякі параметри було введено невірно. Наприклад, поле «firstName» залишимо пустим:

Рисунок 4.2 – Відповідь маршруту POST «users/register», якщо поле «firstName» залишимо пустим

Або «email» введено у неправильному форматі:

Рисунок 4.3 – Відповідь маршруту POST «users/register» при неправильно введеному полі «email»

Далі Спробуємо залогінитись за допомогою маршруту **POST «users/login»**. Як і в попередньому пункті, перевіримо роботу при правильних даних :

Рисунок 4.4 – Відповідь маршруту POST «users/login» при правильних даних

Введемо неіснуючий логін:

Рисунок 4.5 – Відповідь маршруту POST «users/login» при введені неіснуючого ім’я користувача

Введемо невірний пароль інформація зображена на рисунках 4.6. , 4.7.

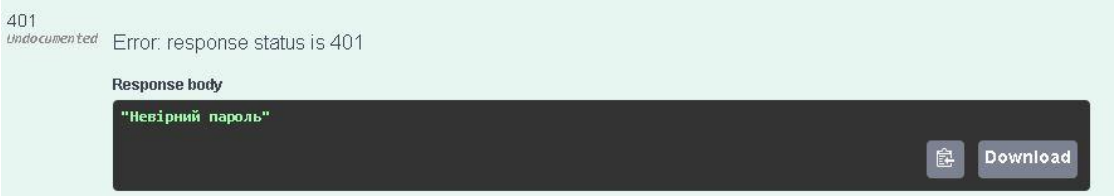


Рисунок 4.6 – Відповідь маршруту POST «users/login» при введені неправильного пароля

Далі спробуємо використати маршрут **POST «persons»**, який додає дані про особу до бази даних

Рисунок 4.7 – Відповідь маршруту POST «persons» при спробі надіслати запит без токену

Як можна побачити вертається 401(Unauthorized) статус код, який свідчить, що користувач не авторизований. Для того, щоб клієнт міг відправити запит та отримати задовольняючу відповідь треба зробити наступне (рис 4.8):

Рисунок 4.8 – Вікно для додавання token у «Authorization» header

Тут ми у header «Authorization» додаємо токен, який буде перевіряти бекенд, щоб по-перше ідентифікувати користувача, а по-друге надати йому певні доступи, залежно від ролі. Знову спробуємо скористатися маршрутом (рис 4.9):

Рисунок 4.9 – Відповідь маршруту POST «persons», при правильно введених даних

Отримали очікувану відповідь(id новоствореної особи). Також перевіримо даний маршрут на відловлювання помилок (рис. 4.10, 4.11, 4.12, 4.13):

Рисунок 4.10 – Відповідь маршруту POST «persons», коли поле «fullName» залишилось пустим

Рисунок 4.11 – Відповідь маршруту POST «persons», якщо у поле «fullName» введено кількість слів, яка не дорівнює 3

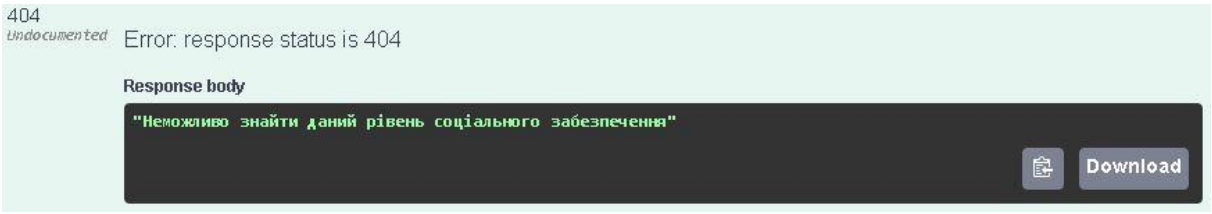


Рисунок 4.12 – Відповідь маршруту POST «persons», якщо запису із «socialSecurityId» не було знайдемо у базі даних, у таблиці рівнів соціального забезпечення

Рисунок 4.13 – Відповідь маршруту POST «persons», якщо запису із «documentTypeId» не було знайдемо у базі даних, у таблиці типів документа

Протестуємо маршрут по додаванню зв’язків між особами зображено на рисунку 4.14, 4.15, 4.16 **POST «persons/{id}/relationships»**:

Рисунок 4.14 – Відповідь маршруту POST «persons/{id}/relationships», при правильно введених даних

Рисунок 4.15 – Відповідь маршруту POST «persons/{id}/relationships», при намаганні додати зв’язок з одною і тою самою особою

Рисунок 4.16 – Відповідь маршруту POST «persons/{id}/relationships», при намаганні додати зв’язок з неіснуючою особою

Рисунок 4.17 – Відповідь маршруту POST «persons/{id}/relationships», при намаганні додати зв’язок неіснуючого рівня

Далі протестуємо маршрут по додаванню події POST«incidents», результат зображено на рисунку 4.18-4.24.

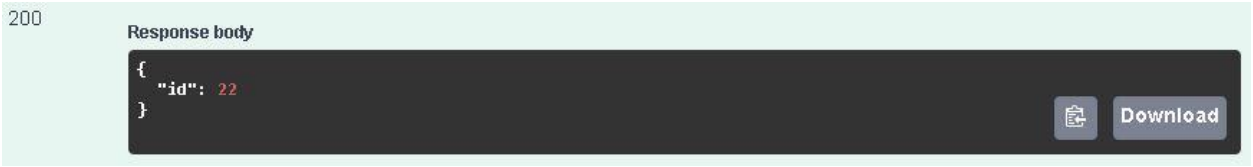


Рисунок 4.18 – Відповідь маршруту POST «incidents», при правильно введених даних

Рисунок 4.19 – Відповідь маршруту POST «incidents», якщо додати подію з реєстраційним номером, який вже є в базі

Рисунок 4.20 – Відповідь маршруту POST «incidents», якщо додати подію з реєстраційним номером, який вже є в базі

Рисунок 4.21 – Відповідь маршруту POST «incidents», якщо додати подію з пустим списком осіб якогось типу(жертва, кривдник)

Рисунок 4.22 – Відповідь маршруту POST «incidents», якщо додати особу одночасно у списки різного типу або додати її двічі у один список

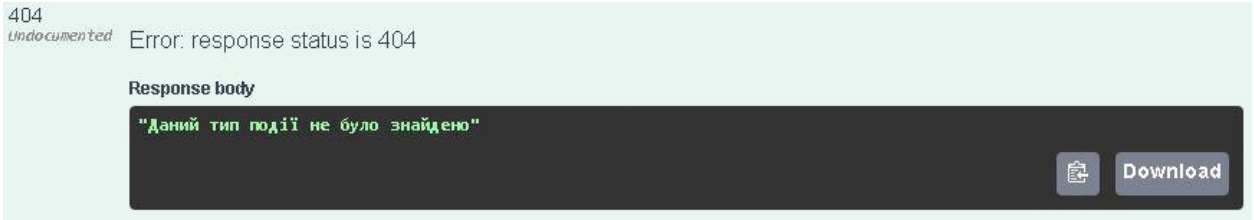


Рисунок 4.23 – Відповідь маршруту POST «incidents», якщо не було знайдено запис у базі даних, у таблиці **Типів Подій** із id, який рівний «incidentTypeId»

Рисунок 4.24 – Відповідь маршруту POST «incidents», якщо не було знайдено запис у базі даних, у таблиці **Кваліфікацій** із id, який рівний «qualificationId»

Подивимося на результати маршрутів по перегляду інформації, а саме GET «incidents/{id}», GET«persons/{id}», GET«users/id» (рис 4.25-4.27):

Рисунок 4.25 – Відповідь маршруту GET «incidents/{id}»

Рисунок 4.26 – Відповідь маршруту GET «persons /{id}»

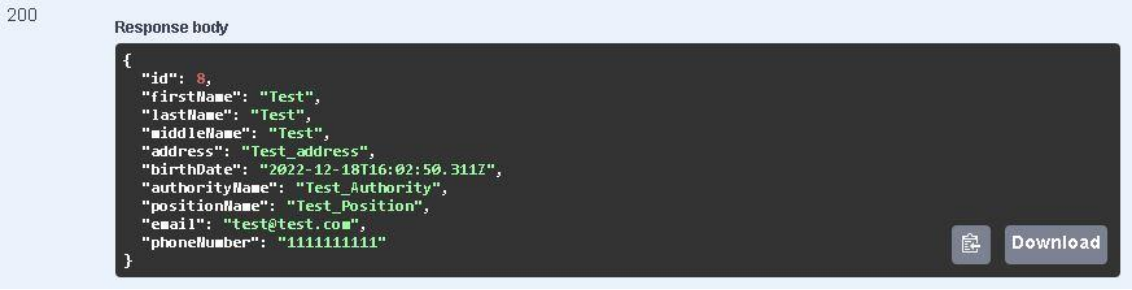


Рисунок 4.27 – Відповідь маршруту GET «users /{id}»

Отже, після тестування модуля підсистеми захисту можна зробити висновок, що він відпрацьовує правильно та виконує всі зазначені функції, а також відловлювання помилок. Авторизація та захист від несанкціонованого користування маршрутами виконується без проблем. Ще можна побачити, що маршрути для перегляду інформації про користувачів, осіб та подій повертають доволі детальну інформацію, якої вистає для виконання технічних вимог. Маршрути по додаванню даних також працюють без нарікань та не дають створити нові об’єкти, якщо у введених даних є логічні і синтаксичні помилки.

5 ЕКОНОМІЧНА ЧАСТИНА

Виконання науково-дослідної роботи завжди передбачає отримання певних результатів і вимагає відповідних витрат. Результати виконаної роботи завжди дають нам нові знання, які в подальшому можуть бути використані для удосконалення та/або розробки (побудови) нових, більш продуктивних зразків техніки, процесів та програмного забезпечення.

Дослідження на тему «Захищена система збирання та аналізування даних для спеціальних задач. Частина 2. Підсистема захисту» може бути віднесено до фундаментальних і пошукових наукових досліджень і спрямоване на вирішення наукових проблем, пов’язаних з практичним застосуванням. Основою таких досліджень є науковий ефект, який виражається в отриманні наукових результатів, які збільшують обсяг знань про природу, техніку та суспільство, які розвивають теоретичну базу в тому чи іншому науковому напрямку, що дозволяє виявити нові закономірності, які можуть використовуватися на практиці.

Для цього випадку виконаємо такі етапи робіт:

1) здійснимо проведення наукового аудиту досліджень, тобто встановлення їх наукового рівня та значимості;

2) проведемо планування витрат на проведення наукових досліджень;

5.1 Оцінювання наукового ефекту

Основними ознаками наукового ефекту науково-дослідної роботи є новизна роботи, рівень її теоретичного опрацювання, перспективність, рівень розповсюдження результатів, можливість реалізації. Науковий ефект НДР на тему «Захищена система збирання та аналізування даних для спеціальних задач. Частина 2. Підсистема захисту» можна охарактеризувати двома показниками: ступенем наукової новизни та рівнем теоретичного опрацювання.

Значення показників ступеня новизни і рівня теоретичного опрацювання науково-дослідної роботи в балах наведені в табл. 5.1 та 5.2.

Таблиця 5.1 – Показники ступеня новизни науково-дослідної роботи виставлені експертами

Ступінь новизни	Характеристика ступеня новизни	Значення ступеня новизни, бали		
		Експерти (ПІБ, посада)		
		1	2	3
Принципово нова	Робота якісно нова за постановкою задачі і ґрунтується на застосуванні оригінальних методів дослідження. Результати дослідження відкривають новий напрям в даній галузі науки і техніки. Отримані принципово нові факти, закономірності; розроблена нова теорія. Створено принципово новий пристрій, спосіб, метод	-	-	-
Нова	Отримана нова інформація, яка суттєво зменшує невизначеність наявних значень (по-новому або вперше пояснені відомі факти, закономірності, впроваджені нові поняття, розкрита структура змісту). Проведено суттєве вдосконалення, доповнення і уточнення раніше досягнутих результатів	56	60	58
Відносно нова	Робота має елементи новизни в постановці задачі і методах дослідження. Результати дослідження систематизують і узагальнюють наявну інформацію, визначають шляхи подальших досліджень; вперше знайдено зв’язок (або знайдено новий зв’язок) між явищами. В принципі відомі положення розповсюджені на велику кількість об’єктів, в результаті чого знайдено ефективне рішення. Розроблені більш прості способи для досягнення відомих результатів. Проведена часткова раціональна модифікація (з ознаками новизни)	-	-	-
Традиційна	Робота виконана за традиційною методикою. Результати дослідження мають інформаційний характер. Підтверджені або поставлені під сумнів відомі факти та твердження, які потребують перевірки. Знайдено новий варіант рішення, який не дає суттєвих переваг в порівнянні з існуючим	-	-	-
Не нова	Отримано результат, який раніше зафіксований в інформаційному полі, та не був відомий авторам	-	-	-
Середнє значення балів експертів		58,0		

Згідно отриманого середнього значення балів експертів ступінь новизни характеризується як нова, тобто отримана нова інформація, яка суттєво зменшує невизначеність наявних знань (по-новому або вперше пояснені відомі факти, закономірності, впроваджені нові поняття, розкрита структура змісту) та проведено суттєве вдосконалення, доповнення і уточнення раніше досягнутих результатів.

Таблиця 5.2 – Показники рівня теоретичного опрацювання науково-дослідної роботи виставлені експертами

Характеристика рівня теоретичного опрацювання	Значення показника рівня теоретичного опрацювання, бали		
	Експерт (ПІБ, посада)		
	1	2	3
Відкриття закону, розробка теорії	-	-	-
Глибоке опрацювання проблеми: багатоаспектний аналіз зв'язків, взаємозалежності між фактами з наявністю пояснень, наукової систематизації з побудовою евристичної моделі або комплексного прогнозу	68	70	66
Розробка способу (алгоритму, програми), пристрою, отримання нової речовини	-	-	-
Елементарний аналіз зв'язків між фактами та наявною гіпотезою, класифікація, практичні рекомендації для окремого випадку тощо	-	-	-
Опис окремих елементарних фактів, викладення досвіду, результатів спостережень, вимірювань тощо	-	-	-
Середнє значення балів експертів	68,0		

Згідно отриманого середнього значення балів експертів рівень теоретичного опрацювання науково-дослідної роботи характеризується як глибоке опрацювання проблеми: багатоаспектний аналіз зв'язків, взаємозалежності між фактами з наявністю пояснень, наукової систематизації з побудовою евристичної моделі або комплексного прогнозу.

Показник, який характеризує рівень наукового ефекту, визначаємо за формулою.

$$E_{нау} = 0,6 \cdot k_{нов} + 0,4 \cdot k_{теор}, \quad (5.1)$$

де $k_{нов}, k_{теор}$ - показники ступеня новизни та рівня теоретичного опрацювання науково-дослідної роботи, $k_{нов} = 58,0, k_{теор} = 68,0$ балів;

0,6 та 0,4 – питома вага (значимість) показників ступеня новизни та рівня теоретичного опрацювання науково-дослідної роботи.

$$E_{нау} = 0,6 \cdot k_{нов} + 0,4 \cdot k_{теор} = 0,6 \cdot 58,0 + 0,4 \cdot 68,00 = 62,00 \text{ балів.}$$

Визначення характеристики показника $E_{нау}$ проводиться на основі висновків експертів виходячи з граничних значень, які наведені в табл. 5.3.

Таблиця 5.3 – Граничні значення показника наукового ефекту

Досягнутий рівень показника	Кількість балів
Високий	70…100
Середній	50…69
Достатній	15…49
Низький (помилкові дослідження)	1…14

Відповідно до визначеного рівня наукового ефекту проведеної науково-дослідної роботи на тему «Захищена система збирання та аналізування даних для спеціальних задач. Частина 2. Підсистема захисту», даний рівень становить 62,00 балів і відповідає статусу – середній рівень. Тобто у даному випадку можна вести мову про потенційну фактичну ефективність науково-дослідної роботи.

5.2 Розрахунок витрат на здійснення науково-дослідної роботи

Витрати, пов’язані з проведенням науково-дослідної роботи на тему «Захищена система збирання та аналізування даних для спеціальних задач. Частина 2. Підсистема захисту», під час планування, обліку і калькулювання собівартості науково-дослідної роботи групуємо за відповідними статтями.

5.2.1 Витрати на оплату праці

До статті «Витрати на оплату праці» належать витрати на виплату основної та додаткової заробітної плати керівникам відділів, лабораторій, секторів і груп, науковим, інженерно-технічним працівникам, конструкторам, технологам, креслярам, копіювальникам, лаборантам, робітникам, студентам, аспірантам та іншим працівникам, безпосередньо зайнятим виконанням конкретної теми, обчисленої за посадовими окладами, відрядними розцінками, тарифними ставками згідно з чинними в організаціях системами оплати праці.

Основна заробітна плата дослідників

Витрати на основну заробітну плату дослідників ($З_о$) розраховуємо у відповідності до посадових окладів працівників, за формулою:

$$З_о = \sum_{i=1}^{k} \frac{M_{ni} \cdot t_i}{T_p}, \qquad (5.2)$$

де k – кількість посад дослідників залучених до процесу досліджень;

M_{ni} – місячний посадовий оклад конкретного дослідника, грн;

t_i – число днів роботи конкретного дослідника, дн.;

T_p – середнє число робочих днів в місяці, T_p=24 дні.

$$З_о = 17400{,}00 \cdot 24 / 24 = 17400{,}00 \ грн.$$

Проведені розрахунки зведемо до таблиці.

Таблиця 5.4 – Витрати на заробітну плату дослідників

Найменування посади	Місячний посадовий оклад, грн	Оплата за робочий день, грн	Число днів роботи	Витрати на заробітну плату, грн
Керівник проекту	17400,00	725,00	24	17400,00
Науковий співробітник	17150,00	714,58	14	10004,17
Інженер-програміст 1-ї категорії	17050,00	710,42	24	17050,00
Лаборант	6750,00	281,25	15	4218,75
Всього				48672,92

Основна заробітна плата робітників

Витрати на основну заробітну плату робітників ($З_р$) за відповідними найменуваннями робіт НДР на тему «Захищена система збирання та аналізування даних для спеціальних задач. Частина 2. Підсистема захисту» розраховуємо за формулою:

$$З_р = \sum_{i=1}^{n} C_i \cdot t_i \,, \tag{5.3}$$

де C_i – погодинна тарифна ставка робітника відповідного розряду, за виконану відповідну роботу, грн/год;

t_i – час роботи робітника при виконанні визначеної роботи, год.

Погодинну тарифну ставку робітника відповідного розряду C_i можна визначити за формулою:

$$C_i = \frac{M_M \cdot K_i \cdot K_c}{T_p \cdot t_{зм}} \,, \tag{4.4}$$

де $М_М$ – розмір прожиткового мінімуму працездатної особи, або мінімальної місячної заробітної плати (в залежності від діючого законодавства), приймемо $Мм$=6700,00 грн;

$К_і$ – коефіцієнт міжкваліфікаційного співвідношення для встановлення тарифної ставки робітнику відповідного розряду (табл. 5.5).

Таблиця 5.5 – Величина витрат на основну заробітну плату робітників

Найменування робіт	Тривалість роботи, год	Розряд роботи	Тарифний коефіцієнт	Погодинна тарифна ставка, грн	Величина оплати на робітника грн
Встановлення допоміжного обладнання	8,50	2	1,10	63,34	538,36
Інсталяція програмного забезпечення	6,25	3	1,35	77,73	485,82
Встановлення цифрових обчислювальних систем	5,20	5	1,70	97,88	508,99
Відлагодження програмних модулів аналізу даних	5,50	4	1,50	86,37	475,02
Підготовка дослідження	8,00	4	1,50	86,37	690,94
Формування бази даних результатів дослідження	14,00	2	1,10	63,34	886,70
Всього					3585,82

$К_с$ – мінімальний коефіцієнт співвідношень місячних тарифних ставок робітників першого розряду з нормальними умовами праці виробничих об’єднань і підприємств до законодавчо встановленого розміру мінімальної заробітної плати.

$Т_р$ – середнє число робочих днів в місяці, приблизно $Т_р$ = 24 дн;

$t_{зм}$ – тривалість зміни, год.

C_1 = 6700,00 · 1,10 · 1,65 / (24 · 8) = 63,34 *грн.*

$З_{р1}$ = 63,34 · 8,50 = 538,36 *грн.*

Додаткова заробітна плата дослідників та робітників

Додаткову заробітну плату розраховуємо як 10 … 12% від суми основної заробітної плати дослідників та робітників за формулою:

$$З_{дод} = (З_о + З_р) \cdot \frac{Н_{дод}}{100\%}, \quad (5.5)$$

де $Н_{дод}$ – норма нарахування додаткової заробітної плати. Приймемо 11%.

$З_{дод}$ = (48672,92 + 3585,82) · 11 / 100% = 5748,46 грн.

5.2.2 Відрахування на соціальні заходи

Нарахування на заробітну плату дослідників та робітників розраховуємо як 22% від суми основної та додаткової заробітної плати дослідників і робітників за формулою:

$$З_н = (З_о + З_р + З_{дод}) \cdot \frac{Н_{зп}}{100\%} \quad (5.6)$$

де $Н_{зп}$ – норма нарахування на заробітну плату. Приймаємо 22%.

Зн = (48672,92 + 3585,82 + 5748,46) · 22 / 100% = 12761,58 грн.

5.2.3 Сировина та матеріали

До статті «Сировина та матеріали» належать витрати на сировину, основні та допоміжні матеріали, інструменти, пристрої та інші засоби і предмети праці, які придбані у сторонніх підприємств, установ і організацій та витрачені на проведення досліджень за темою «Захищена система збирання та аналізування даних для спеціальних задач. Частина 2. Підсистема захисту».

Витрати на матеріали на даному етапі проведення досліджень в основному пов’язані з використанням моделей елементів та моделювання роботи і

досліджень за допомогою комп'ютерної техніки та створення експериментальних математичних моделей або програмного забезпечення, тому дані витрати формуються на основі витратних матеріалів характерних для офісних робіт.

Витрати на матеріали (*М*), у вартісному вираженні розраховуються окремо по кожному виду матеріалів за формулою:

$$M = \sum_{j=1}^{n} H_j \cdot Ц_j \cdot К_j - \sum_{j=1}^{n} B_j \cdot Ц_{вj} , \qquad (5.7)$$

де H_j – норма витрат матеріалу *j*-го найменування, кг;

n – кількість видів матеріалів;

$Ц_j$ – вартість матеріалу *j*-го найменування, грн/кг;

$К_j$ – коефіцієнт транспортних витрат, ($К_j$ = *1,1 ... 1,15*);

B_j – маса відходів *j*-го найменування, кг;

$Ц_{вj}$ – вартість відходів *j*-го найменування, грн/кг.

M_1 = *2,0 · 225,00 · 1,11 – 0 · 0 = 499,50 грн.*

Витрати на комплектуючі *($К_в$)*, які використовують при проведенні НДР на тему «Захищена система збирання та аналізування даних для спеціальних задач. Частина 2. Підсистема захисту», розраховуємо, згідно з їхньою номенклатурою, за формулою:

Проведені розрахунки зведемо до таблиці 5.6.

Таблиця 5.6 – Витрати на матеріали

Найменування матеріалу, марка, тип, сорт	Ціна за 1 кг, грн	Норма витрат, кг	Величина відходів, кг	Ціна відходів, грн/кг	Вартість витраченого матеріалу, грн
Папір канцелярський офісний (А4)	225,00	2,0	-	-	499,50
Папір для заміток (А5)	116,00	4,0	-	-	515,04
Начиння канцелярське	195,00	3,0	-	-	649,35
Органайзер офісний	183,00	3,0	-	-	609,39
Картридж для принтера	950,00	1,0	-	-	1054,50
Диск оптичний CD-RW	21,00	3,0	-	-	69,93
USB флеш накопичувач Transcend 16Gb JetFlash 700 (TS64GJF700)	120,00	1,0	-	-	133,20
Всього					3530,91

5.2.4 Розрахунок витрат на комплектуючі

Витрати на комплектуючі *($К_в$)*, які використовують при проведенні НДР на тему «Захищена система збирання та аналізування даних для спеціальних задач. Частина 2. Підсистема захисту», розраховуємо, згідно з їхньою номенклатурою, за формулою:

$$К_в = \sum_{j=1}^{n} Н_j \cdot Ц_j \cdot К_j \qquad (5.8)$$

де H_j – кількість комплектуючих *j*-го виду, шт.;

$Ц_j$ – покупна ціна комплектуючих *j*-го виду, грн;

$К_j$ – коефіцієнт транспортних витрат, ($К_j$ = *1,1 … 1,15*).

$К_в = 1 \cdot 3079{,}00 \cdot 1{,}11 = 3417{,}69$ *грн.*

Проведені розрахунки зведемо до таблиці.

Таблиця 5.7 – Витрати на комплектуючі

Найменування комплектуючих	Кількість, шт.	Ціна за штуку, грн	Сума, грн
Зовнішній жорсткий диск 2.5" 2TB Seagate (STGD2000200)	1	3079,00	3417,69
Концентратор Defender SEPTIMA SLIM (83505)	1	400,00	444,00
Кабель для передачі даних USB to COM 1.0m Patron (CAB-PN-USB-COM)	1	354,00	392,94
Всього			4254,63

5.2.5 Спецустаткування для наукових (експериментальних) робіт

До статті «Спецустаткування для наукових (експериментальних) робіт» належать витрати на виготовлення та придбання спецустаткування необхідного для проведення досліджень, також витрати на їх проектування, виготовлення, транспортування, монтаж та встановлення.

Балансову вартість спецустаткування розраховуємо за формулою:

$$B_{спец} = \sum_{i=1}^{k} Ц_i \cdot C_{пр.i} \cdot K_i \ , \qquad (5.9)$$

де $Ц_i$ – ціна придбання одиниці спецустаткування даного виду, марки, грн;

$C_{пр.і}$ –кількість одиниць устаткування відповідного найменування, які придбані для проведення досліджень, шт.;

$K_і$ – коефіцієнт, що враховує доставку, монтаж, налагодження устаткування тощо, ($K_і$ = *1,10…1,12);*

k – кількість найменувань устаткування.

$B_{спец}$ = *2999,00 · 1 · 1,11 = 3328,89 грн.*

Отримані результати зведемо до таблиці:

Таблиця 4.8 – Витрати на придбання спецустаткування по кожному виду

Найменування устаткування	Кількість, шт	Ціна за одиницю, грн	Вартість, грн
Маршрутизатор Mikrotik hAP ac (RBD52G-5HacD2HnD-TC)	1	2999,00	3328,89
Всього			3328,89

5.2.6 Програмне забезпечення для наукових (експериментальних) робіт

До статті «Програмне забезпечення для наукових (експериментальних) робіт» належать витрати на розробку та придбання спеціальних програмних засобів і програмного забезпечення, (програм, алгоритмів, баз даних) необхідних для проведення досліджень, також витрати на їх проектування, формування та встановлення.

Балансову вартість програмного забезпечення розраховуємо за формулою:

$$B_{прг} = \sum_{i=1}^{k} Ц_{iпрг} \cdot C_{прг.і} \cdot K_i \ , \qquad (5.10)$$

де $Ц_{iпрг}$ – ціна придбання одиниці програмного засобу даного виду, грн;

$C_{прг.і}$ – кількість одиниць програмного забезпечення відповідного найменування, які придбані для проведення досліджень, шт.;

$K_і$ – коефіцієнт, що враховує інсталяцію, налагодження програмного засобу тощо, ($K_і = 1,10…1,12$);

k – кількість найменувань програмних засобів.

$B_{прг} = 4280,00 \cdot 1 \cdot 1,11 = 4750,80$ *грн.*

Отримані результати зведемо до таблиці:

Таблиця 5.9 – Витрати на придбання програмних засобів по кожному виду

Найменування програмного засобу	Кількість, шт	Ціна за одиницю, грн	Вартість, грн
Прикладне програмне забезпечення розробки системи захисту	1	4280,00	4750,80
Всього			4750,80

5.2.7 Амортизація обладнання, програмних засобів та приміщень

В спрощеному вигляді амортизаційні відрахування по кожному виду обладнання, приміщень та програмному забезпеченню тощо, розраховуємо з використанням прямолінійного методу амортизації за формулою:

$$A_{обл} = \frac{Ц_б}{T_в} \cdot \frac{t_{вик}}{12}, \quad (5.11)$$

де $Ц_б$ – балансова вартість обладнання, програмних засобів, приміщень тощо, які використовувались для проведення досліджень, грн;

$t_{вик}$ – термін використання обладнання, програмних засобів, приміщень під час досліджень, місяців;

$T_в$ – строк корисного використання обладнання, програмних засобів, приміщень тощо, років.

$A_{обл} = (36499{,}00 \cdot 1) / (2 \cdot 12) = 1520{,}79$ *грн.*

Проведені розрахунки зведемо до таблиці.

Таблиця 5.10 – Амортизаційні відрахування по кожному виду обладнання

Найменування обладнання	Балансова вартість, грн	Строк корисного використання, років	Термін використання обладнання, місяців	Амортизаційні відрахування, грн
Ноутбук Lenovo Thinkpad T14 Gen 2, оперативна пам'ять 16GB, процесор Intel core i7-1185G7, пам'ять на жорсткому диску 256 GB	36499,00	2	1	1520,79
Робоче місце інженера-розробника ПЗ	8752,00	5	1	145,87
Пристрої передачі даних	6810,00	2	1	283,75
Пристрій виводу інформації	6791,00	5	1	113,18
Оргтехніка	8300,00	4	1	172,92
Приміщення лабораторії	500000,00	20	1	2083,33
ОС Windows 10	8360,00	2	1	348,33
Прикладний пакет Microsoft Office 2016	7900,00	2	1	329,17
Всього				4997,34

5.2.8 Паливо та енергія для науково-виробничих цілей

Витрати на силову електроенергію (B_e) розраховуємо за формулою:

$$B_e = \sum_{i=1}^{n} \frac{W_{yi} \cdot t_i \cdot Ц_e \cdot K_{вni}}{\eta_i}, \quad (5.12)$$

де W_{yi} – встановлена потужність обладнання на визначеному етапі розробки, кВт;

t_i – тривалість роботи обладнання на етапі дослідження, год;

Таблиця 5.11 – Витрати на електроенергію

Найменування обладнання	Встановлена потужність, кВт	Тривалість роботи, год	Сума, грн
Ноутбук Lenovo Thinkpad T14 Gen 2, оперативна пам'ять 16GB, процесор Intel core i7-1185G7, пам'ять на жорсткому диску 256 GB	0,04	180,0	44,64
Робоче місце інженера-розробника ПЗ	0,10	180,0	111,60
Пристрої передачі даних	0,01	180,0	11,16
Пристрій виводу інформації	0,50	10,0	31,00
Оргтехніка	0,62	2,0	7,69
Маршрутизатор Mikrotik hAP ac (RBD52G-5HacD2HnD-TC)	0,01	180,0	11,16
Всього			217,25

$Ц_е$ – вартість 1 кВт-години електроенергії, грн; (вартість електроенергії визначається за даними енергопостачальної компанії), приймемо $Ц_е$ = 6,20 грн;

$К_{вп і}$ – коефіцієнт, що враховує використання потужності, $К_{вп і} < 1$;

$\eta_і$ – коефіцієнт корисної дії обладнання, $\eta_і < 1$.

$В_е = 0,04 \cdot 180,0 \cdot 6,20 \cdot 0,95 / 0,97 = 44,64$ *грн.*

Проведені розрахунки зведемо до таблиці.

5.2.9 Службові відрядження

До статті «Службові відрядження» дослідної роботи на тему «Захищена система збирання та аналізування даних для спеціальних задач. Частина 2. Підсистема захисту» належать витрати на відрядження штатних працівників, працівників організацій, які працюють за договорами цивільно-правового характеру, аспірантів, зайнятих розробленням досліджень, відрядження, пов’язані з проведенням випробувань машин та приладів, а також витрати на відрядження на наукові з’їзди, конференції, наради, пов’язані з виконанням конкретних досліджень.

Витрати за статтею «Службові відрядження» розраховуємо як 20…25% від суми основної заробітної плати дослідників та робітників за формулою:

$$В_{св} = (З_о + З_р) \cdot \frac{Н_{св}}{100\%}, \qquad (5.13)$$

де $Н_{св}$ – норма нарахування за статтею «Службові відрядження», приймемо $Н_{св}$ = 20%.

$В_{св} = (48672,92 + 3585,82) \cdot 20 / 100\% = 10451,75$ *грн.*

5.2.10 Витрати на роботи, які виконують сторонні підприємства, установи і організації

Витрати за статтею «Витрати на роботи, які виконують сторонні підприємства, установи і організації» розраховуємо як 30…45% від суми основної заробітної плати дослідників та робітників за формулою:

$$B_{сп} = (З_о + З_р) \cdot \frac{Н_{сп}}{100\%}, \quad (5.14)$$

де $Н_{сп}$ – норма нарахування за статтею «Витрати на роботи, які виконують сторонні підприємства, установи і організації», приймемо $Н_{сп}$= 30%.

$B_{сп}$ = (48672,92 + 3585,82) · 30 / 100% = 15677,62 грн.

5.2.11 Інші витрати

До статті «Інші витрати» належать витрати, які не знайшли відображення у зазначених статтях витрат і можуть бути віднесені безпосередньо на собівартість досліджень за прямими ознаками.

Витрати за статтею «Інші витрати» розраховуємо як 50…100% від суми основної заробітної плати дослідників та робітників за формулою:

$$І_в = (З_о + З_р) \cdot \frac{Н_{ів}}{100\%}, \quad (5.15)$$

де $Н_{ів}$ – норма нарахування за статтею «Інші витрати», приймемо $Н_{ів}$ = 60%.

$І_в$ = (48672,92 + 3585,82) · 60 / 100% = 31355,24 грн.

5.2.12 Накладні (загальновиробничі) витрати

До статті «Накладні (загальновиробничі) витрати» належать: витрати, пов’язані з управлінням організацією; витрати на винахідництво та раціоналізацію; витрати на підготовку (перепідготовку) та навчання кадрів; витрати, пов’язані з набором робочої сили; витрати на оплату послуг банків; витрати, пов’язані з освоєнням виробництва продукції; витрати на науково-технічну інформацію та рекламу та ін.

Витрати за статтею «Накладні (загальновиробничі) витрати» розраховуємо як 100…150% від суми основної заробітної плати дослідників та робітників за формулою:

$$B_{нзв} = (З_о + З_р) \cdot \frac{Н_{нзв}}{100\%}\ , \qquad (5.16)$$

де $Н_{нзв}$ – норма нарахування за статтею «Накладні (загальновиробничі) витрати», приймемо $Н_{нзв}$ = 130%.

$В_{нзв}$ = (48672,92 + 3585,82) · 130 / 100% = 67936,36 грн.

Витрати на проведення науково-дослідної роботи на тему «Захищена система збирання та аналізування даних для спеціальних задач. Частина 2. Підсистема захисту» розраховуємо як суму всіх попередніх статей витрат за формулою:

$$B_{заг} = З_о + З_р + З_{дод} + З_н + М + К_в + В_{спец} + В_{прг} + А_{обл} + В_е + В_{св} + В_{сп} + І_в + В_{нзв}\ . \qquad (4.17)$$

$В_{заг}$= 48672,92 +3585,82 +5748,46 +12761,58 +3530,91 +4254,63 +3328,89 + 4750,80 + 4997,34 +217,25 +10451,75 +15677,62 +31355,24 +67936,36 = 217269,58 грн.

Загальні витрати *ЗВ* на завершення науково-дослідної (науково-технічної) роботи та оформлення її результатів розраховується за формулою:

$$ЗВ = \frac{B_{заг}}{\eta}\ , \qquad (5.18)$$

де η – коефіцієнт, який характеризує етап (стадію) виконання науково-дослідної роботи, приймемо η=0,95.

ЗВ = 217269,58 / 0,95 = 228704,82 грн.

5.3 Оцінювання важливості та наукової значимості науково-дослідної роботи

Оцінювання та доведення ефективності виконання науково-дослідної роботи фундаментального чи пошукового характеру є достатньо складним процесом і часто базується на експертних оцінках, тому має вірогідний характер.

Для обґрунтування доцільності виконання науково-дослідної роботи на тему «Захищена система збирання та аналізування даних для спеціальних задач. Частина 2. Підсистема захисту» використовується спеціальний комплексний показник, що враховує важливість, результативність роботи, можливість впровадження її результатів у виробництво, величину витрат на роботу.

Комплексний показник K_P рівня науково-дослідної роботи може бути розрахований за формулою:

$$K_P = \frac{I^n \cdot T_C \cdot R}{B \cdot t}, \tag{5.19}$$

де I – коефіцієнт важливості роботи. Приймемо $I = 4$;

n – коефіцієнт використання результатів роботи; $n = 0$, коли результати роботи не будуть використовуватись; $n = 1$, коли результати роботи будуть використовуватись частково; $n = 2$, коли результати роботи будуть використовуватись в дослідно-конструкторських розробках; $n = 3$, коли результати можуть використовуватись навіть без проведення дослідно-конструкторських розробок. Приймемо $n = 2$;

T_C – коефіцієнт складності роботи. Приймемо $T_C = 3$;

R – коефіцієнт результативності роботи; якщо результати роботи плануються вище відомих, то $R = 4$; якщо результати роботи відповідають

відомому рівню, то $R=3$; якщо нижче відомих результатів, то $R=1$. Приймемо $R=4$;

B – вартість науково-дослідної роботи, тис. грн. Приймемо B =228704,82 грн;

t – час проведення дослідження. Приймемо t = 0,08 років, (1 міс.).

Визначення показників I, n, T_C, R, B, t здійснюється експертним шляхом або на основі нормативів.

$$K_P = \frac{I^n \cdot T_C \cdot R}{B \cdot t} = 4^2 \cdot 3 \cdot 4 \,/\, 228{,}7 \cdot 0{,}08 = 10{,}07.$$

Якщо $K_P > 1$, то науково-дослідну роботу на тему «Захищена система збирання та аналізування даних для спеціальних задач. Частина 2. Підсистема захисту» можна вважати ефективною з високим науковим, технічним і економічним рівнем.

Таким чином, уданому розділ було проведено економічне тестування та витрати на проведення науково-дослідної роботи на тему «Захищена система збирання та аналізування даних для спеціальних задач. Частина 2. Підсистема захисту» складають 228704,82 грн. Відповідно до проведеного аналізу та розрахунків рівень наукового ефекту проведеної науково-дослідної роботи на тему «Захищена система збирання та аналізування даних для спеціальних задач. Частина 2. Підсистема захисту» є середній, а дослідження актуальними, рівень доцільності виконання науково-дослідної роботи $K_P > 1$, що свідчить про потенційну ефективність з високим науковим, технічним і економічним рівнем.

ВИСНОВКИ

Проаналізовано та вирішено актуальну науково-прикладну задачу, пов’язану з підвищенням рівня захисту інформації від кібервпливів в комп’ютерних мережах та системах об’єктів захисту, шляхом розробки відповідного методу та засобів захисту інформації.

При вирішенні цієї задачі отримані такі основні результати: Проаналізовано сучасні методи та засоби захисту інформації в комп’ютерних мережах та системах.

Однак, незважаючи на значну кількість підходів до вирішення даної проблеми, вона залишається актуальною не тільки для України, але і для всієї світової спільноти.

Розроблено комбінований метод розпізнавання кіберзагроз інформаційній безпеці комп’ютерних мереж та систем об’єктів захисту, який за рахунок поєднання сигнатурного методу евристичного методу та методу міжмережевих екранів , дозволяє розширити спектр виявлених кіберзагроз.

Розроблено метод захисту веб-застосунків з подальшим удосконаленням і якщо, чітко дотримуватись рекомендацій та послідовність дій, завдяки методу захисту від загроз безпеці веб-додактів можливо захистити веб-застосунок який було розроблено на вимогу головного управління кіберполіції від широко поширених сьогодні хакерів і зловмисників. Звичайно, цей метод буде продовжувати вдосконалюватися й оновлюватися перед появою нових вразливостей і переглядів.

Також після тестування модуля підсистеми захисту можна зробити висновок, що він відпрацьовує правильно та виконує всі зазначені функції, а також відловлювання помилок. Авторизація та захист від несанкціонованого користування маршрутами виконується без проблем. Ще можна побачити, що маршрути для перегляду інформації про користувачів, осіб та подій повертають доволі детальну інформацію, якої вистає для виконання технічних вимог.

Маршрути по додаванню даних також працюють без нарікань та не дають створити нові об’єкти, якщо у введених даних є логічні і синтаксичні помилки

Витрати на проведення науково-дослідної роботи на тему «Захищена система збирання та аналізування даних для спеціальних задач. Частина 2. Підсистема захисту» складають 228704,82 грн. Відповідно до проведеного аналізу та розрахунків рівень наукового ефекту проведеної науково-дослідної роботи на тему «Захищена система збирання та аналізування даних для спеціальних задач. Частина 2. Підсистема захисту» є середній, а дослідження актуальними, рівень доцільності виконання науково-дослідної роботи $K_P > 1$, що свідчить про потенційну ефективність з високим науковим, технічним і економічним рівнем.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Захищена система аналізування даних для спеціальних задач. С. І. Смолявський, В. В. Лукічов, І. О. Волокітенко, матеріали конференції "Контроль і управління в складних системах (КУСС-2022)" ВНТУ, Вінниця, 15-17 листопада 2022 р. [Електронний ресурс]. URL: https://conferences.vntu.edu.ua/index.php/mccs/mccs2022/paper/view/16480 (дата звернення: 16.10.2022)
2. Simultaneous Authentication of Equals: A Secure, Password-Based Key Exchange for Mesh Networks [Електронний ресурс] // IEEE. – 2008. – Ре-жим доступу до ресурсу: https://ieeexplore.ieee.org/document/4622764/.
3. Лужецький В. А. Інформаційна безпека : навчальний посібник – В. А. Лужецький, О. П. Войтович, А. В. Дудатьєв – Вінниця : УНІВЕРСУМВінниця, 2009. – 240 с.
4. Соколов А. В. Защита от комп’ютерного терроризма / Соколов А. В., Степанюк О. М. – СПб: БХВ-Петербург, Арлей, 2002. – 496 с. URL: http://www.dut.edu.ua/ru/lib/1/category/1291/view/812 (дата звернення: 23.04.2021)
5. NooTron [Електронний ресурс] – Режим доступу до ресурсу: https://nootron.net.ua/.
6. В чем разница между режимами WPA-Personal и WPA-Enterprise [Електронний ресурс] // Tp-link. – 2013. – Режим доступу до ресурсу: https://www.tp-link.com/ru/faq-500.html.
7. Атаки на беспроводные сети. Часть 1. 2015. URL: https://www.securitylab.ru/analytics/216360.php (дата звернення: 11.11.2022).
8. Методы и средства защиты информации: підручник. в 2 т. Ленков С. В. Арий, 2008. URL: https://altairbook.com/books/4489361-

metodyisredstva-zashchity-informacii-komplekt-iz-2-knig.html (дата звернення: 22.10.2022).

9. Cross Site Scripting Prevention Cheat Sheet веб-сайт URL https://cheatsheetseries.owasp.org/cheatsheets/Cross_Site_Scripting_Prevention_Cheat_Sheet.html (дата звертання 23.09.2022).
10. OWASP Top Ten URL https://owasp.org/www-project-top-ten/ (дата звертання 27.09.2022).
11. Authentication Cheat Sheet OWASP URL: https://cheatsheetseries.owasp.org/cheatsheets/Authentication_Cheat_Sheet.htm (дата звертання 31.09.2022).
12. OWASP Secure Headers Project URL: https://owasp.org/www-project-secure-headers/ (дата звертання 02.10.2022).
13. Кавецький В. В. Економічне обґрунтування інноваційних рішень: практикум / В. В. Кавецький, В. О. Козловський, І. В. Причепа – Вінниця : ВНТУ, 2016. – 113 с.
14. Методичні вказівки до виконання економічної частини магістерських кваліфікаційних робіт / Уклад. : В. О. Козловський, О. Й. Лесько, В. В. Кавецький. – Вінниця : ВНТУ, 2021. – 42 с.

ДОДАТКИ

95

Додаток А

Протокол перевірки на наявність плагіату

ПРОТОКОЛ ПЕРЕВІРКИ МАГІСТЕРСЬКОЇ КВАЛІФІКАЦІЙНОЇ РОБОТИ НА НАЯВНІСТЬ ТЕКСТОВИХ ЗАПОЗИЧЕНЬ

Назва роботи: Захищена система збирання та аналізування даних для спеціальних задач. Частина 2. Підсистема захисту.

Автор роботи: Смолявський Ілля Сергійович

Тип роботи: магістерська кваліфікаційна робота

Підрозділ кафедра захисту інформації ФІТКІ

(кафедра, факультет)

Показники звіту подібності Unicheck

Оригінальність – 98,1%. Схожість – 1,9%.

Аналіз звіту подібності (відмітити потрібне):

√ 1. Запозичення, виявлені у роботі, оформлені коректно і не містять ознак плагіату.

☐ 2. Виявлені у роботі запозичення не мають ознак плагіату, але їх надмірна кількість викликає сумніви щодо цінності роботи і відсутності самостійності її виконання автором. Роботу направити на розгляд експертної комісії кафедри.

☐ 3. Виявлені у роботі запозичення є недобросовісними і мають ознаки плагіату та/або в ній містяться навмисні спотворення тексту, що вказують на спроби приховування недобросовісних запозичень.

Особа, відповідальна за перевірку _____ (підпис) Каплун В. А. (прізвище, ініціали)

Ознайомлені з повним звітом подібності, який був згенерований системою Unicheck щодо роботи.

Автор роботи _____ (підпис) Смолявський І. С. (прізвище, ініціали)

Керівник роботи _____ (підпис) Лукічов В. В. (прізвище, ініціали)

Додаток Б
Код програмного додатку

UsersController.cs

```
using AutoMapper;
using DomesticAbuseRegistry.Common.Utils.Interfaces;
using DomesticAbuseRegistry.Data.Contexts;
using DomesticAbuseRegistry.Data.Entities;
using DomesticAbuseRegistry.Models.Users;
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Identity;
using Microsoft.AspNetCore.Identity.EntityFrameworkCore;
using Microsoft.AspNetCore.Mvc;
using Microsoft.EntityFrameworkCore;
using Serilog;

namespace DomesticAbuseRegistry.Controllers
{
    [Route("users/[action]")]
    [ApiController]
    public class UsersController : ControllerBase
    {
        private readonly RegistryContext _context;
        private readonly UserManager<User> _userManager;
        private readonly IMapper _mapper;
        private readonly IJwtUtils _jwtUtils;

        public UsersController(RegistryContext context, UserManager<User> userManager,
            IJwtUtils jwtUtils, IMapper mapper)
        {
            _context = context;
            _userManager = userManager;
            _jwtUtils = jwtUtils;
            _mapper = mapper;
        }

        [HttpPost, ActionName("login")]
        public async Task<IActionResult> LoginAsync([FromBody] UserLoginRequest loginRequest)
        {
            try
            {
                var user = await _userManager.FindByEmailAsync(loginRequest.UserName);

                if(user == null)
                {
                    Log.Error("User not found. Username = {0}", loginRequest.UserName);

                    var notFoundResult = new JsonResult("Користувача не знайдено")
                    {
                        StatusCode = 404
                    };
                    return notFoundResult;
                }

                var result = await _userManager.CheckPasswordAsync(user, loginRequest.Password);

                if(!result)
                {
                    Log.Error("User entered incorrect password. UserId = {0}",
                        user.Id);

                    var incorrectPasswordResult = new JsonResult("Невірний пароль")
                    {
                        StatusCode = 401
                    };
                    return incorrectPasswordResult;
                }

                Log.Information("User successfully logged in. UserId = {0}",
                    user.Id);
```

```
                var userInfo = _mapper.Map<UserInfoResponse>(user);

                var loginResponse = new UserLoginResponse
                {
                    UserId = user.Id,
                    Token = _jwtUtils.GenerateToken(userInfo)
                };

                Log.Information("Token for User is generated. UserId = {0}",
                    user.Id);

                return new JsonResult(loginResponse);
            }
            catch(Exception ex)
            {
                Log.Error(ex, "Unexpected exception during request");
            }

            var dbInternalError = new JsonResult("Помилка бази даних")
            {
                StatusCode = 500
            };
            return dbInternalError;
        }

        [HttpPost, ActionName("register")]
        public async Task<IActionResult> RegisterAsync([FromBody] UserRegistryRequest
registryRequest)
        {
            try
            {
                var user = new User
                {
                    UserName = registryRequest.Email,
                    Email = registryRequest.Email,
                    PhoneNumber = registryRequest.PhoneNumber,
                    FirstName = registryRequest.FirstName,
                    MiddleName = registryRequest.MiddleName,
                    LastName = registryRequest.LastName,
                    Address = registryRequest.Address,
                    BirthDate = registryRequest.BirthDate,
                    AuthoritryName = registryRequest.AuthorityName,
                    PositionName = registryRequest.PositionName
                };

                var result = await _userManager.CreateAsync(user, registryRequest.Password);

                if(!result.Succeeded)
                {
                    Log.Error("Failed to register new User. Errors: [{0}]",
                        string.Join("; ", result.Errors.Select(e => e.Description)));

                    var registrationErrorResult = new JsonResult(string.Join(" ",
result.Errors.Select(e => e.Description)))
                    {
                        StatusCode = 400
                    };
                    return registrationErrorResult;
                }

                Log.Information("New user is successfully registered. UserId = {0}",
                    user.Id);

                await _context.SaveChangesAsync();

                var userInfoResponse = _mapper.Map<UserInfoResponse>(await
_userManager.FindByEmailAsync(registryRequest.Email));
                var token = _jwtUtils.GenerateToken(userInfoResponse);

                return new JsonResult(new {User = userInfoResponse, Token = token});
            }
            catch(Exception ex)
            {
                Log.Error(ex, "Unexpected exception during request");
            }

            var dbInternalError = new JsonResult("Помилка бази даних")
```

```
            {
                StatusCode = 500
            };
            return dbInternalError;
        }
```

PersonsController.cs

```
using AutoMapper;
using AutoMapper.Execution;
using DomesticAbuseRegistry.Data.Contexts;
using DomesticAbuseRegistry.Data.Entities;
using DomesticAbuseRegistry.Models.Incidents;
using DomesticAbuseRegistry.Models.Persons;
using DomesticAbuseRegistry.Models.Users;
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Http;
using Microsoft.AspNetCore.Http.Features;
using Microsoft.AspNetCore.Identity;
using Microsoft.AspNetCore.Mvc;
using Microsoft.EntityFrameworkCore;
using Microsoft.VisualBasic;
using Serilog;

namespace DomesticAbuseRegistry.Controllers
{
    [Route("persons/[action]")]
    [ApiController]
    [Authorize]
    public class PersonsController : ControllerBase
    {
        private readonly RegistryContext _context;
        private readonly IMapper _mapper;

        public PersonsController(RegistryContext context, IMapper mapper)
        {
            _context = context;
            _mapper = mapper;
        }
       [HttpPost, ActionName("")]
        public async Task<IActionResult> AddPersonAsync([FromBody] PersonAddRequest
personAddRequest)
        {
            try
            {
                var personNames = personAddRequest.FullName.Split(' ');
                if(personNames.Length != 3)
                {
                    Log.Error("Fullname is invalid. Fullname = {0}",
                        personAddRequest.FullName);

                    var badRequestResult = new JsonResult("Невірно введено ПІБ")
                    {
                        StatusCode = 400
                    };
                    return badRequestResult;
                }

                if (personAddRequest.SocialSecurityId != null)
                {
                    var socialSecurity = await _context.SocialSecurities
                        .FirstOrDefaultAsync(s => s.Id == personAddRequest.SocialSecurityId);
                    if(socialSecurity == null)
                    {
                        Log.Error("Social Security is not found. Social security Id = {0}",
                            personAddRequest.SocialSecurityId);

                        var notFoundResult = new JsonResult("Неможливо знайти даний рівень
соціального забезпечення")
                        {
                            StatusCode = 404
                        };
                        return notFoundResult;
                    }
                }

                var person = new Person();
                _mapper.Map(personAddRequest, person);
```

```
                person.LastName = personNames[0];
                person.FirstName = personNames[1];
                person.MiddleName = personNames[2];

                if(personAddRequest.DocumentTypeId != null &&
                    personAddRequest.IssuingAuthority != null &&
                    personAddRequest.DocumentNumber != null &&
                    personAddRequest.DocumentIssueDate != null &&
                    !string.IsNullOrEmpty(personAddRequest.DocumentSeries))
                {
                    var personDocumentType = await _context.PersonDocumentTypes
                        .FirstOrDefaultAsync(s => s.Id == personAddRequest.DocumentTypeId);
                    if(personDocumentType == null)
                    {
                        Log.Error("Person document type is not found. Person document type Id =
{0}",
                            personAddRequest.SocialSecurityId);

                        var notFoundResult = new JsonResult("Неможливо знайти даний тип документу
особи")
                        {
                            StatusCode = 404
                        };
                        return notFoundResult;
                    }

                    person.PersonDocument = new PersonDocument
                    {
                        PersonDocumentTypeId = (uint)personAddRequest.DocumentTypeId,
                        Number = (uint)personAddRequest.DocumentNumber,
                        IssuingAuthority = (uint)personAddRequest.IssuingAuthority,
                        Series = personAddRequest.DocumentSeries,
                        IssueDate = (DateTime)personAddRequest.DocumentIssueDate
                    };
                }

                await _context.Persons.AddAsync(person);
                await _context.SaveChangesAsync();

                Log.Information("Person was successfully added to Db. " +
                    "Person Id = {0}", person.Id);

                return new JsonResult(person.Id);
            }
            catch(Exception ex)
            {
                Log.Error(ex, "Unexpected exception during request");
            }

            var dbInternalError = new JsonResult("Помилка бази даних")
            {
                StatusCode = 500
            };
            return dbInternalError;
        }

        [HttpPost, ActionName("{id}/relationships")]
        public async Task<IActionResult> AddPersonRelationshipsAsync([FromRoute] uint id, [FromBody]
PersonAddRelationshipsRequest personAddRelationshipsRequests)
        {
            try
            {
                var person = await _context.Persons.FirstOrDefaultAsync(p => p.Id == id);

                if(person == null)
                {
                    Log.Error("Person was not found. PerosnId = {0}", id);

                    var notFoundResult = new JsonResult("Дану особу не було знайдено")
                    {
                        StatusCode = 404
                    };
                    return notFoundResult;
                }

                foreach(var relationship in personAddRelationshipsRequests.personRelationships)
                {
```

```
                    if(relationship.RelationshipPersonId == id)
                    {
                        Log.Error("Person can not have relationship with itself. PerosnId = {0}",
id);

                        var badRequestResult = new JsonResult("Особа не може перебувати у відносинах
сама з собою")
                        {
                            StatusCode = 400
                        };
                        return badRequestResult;
                    }

                    var relationshipPerson = await _context.Persons
                        .FirstOrDefaultAsync(p => p.Id == relationship.RelationshipPersonId);

                    if(relationshipPerson == null)
                    {
                        Log.Error("Person was not found. PerosnId = {0}", id);

                        var notFoundResult = new JsonResult("Дану особу не було знайдено")
                        {
                            StatusCode = 404
                        };
                        return notFoundResult;
                    }

                    var relationshipType = await _context.RelationshipTypes
                        .FirstOrDefaultAsync(rt => rt.Id == relationship.RelationshipTypeId);

                    if(relationshipType == null)
                    {
                        Log.Error("Given Relationship type was not found. " +
                            "RelationshipTypeId = {0}", relationship.RelationshipTypeId);

                        var notFoundResult = new JsonResult("Даного рівень зв'язку не було
знайдено")
                        {
                            StatusCode = 404
                        };
                        return notFoundResult;
                    }

                    var relationshipToAdd = new Relationship
                    {
                        FirstPersonId = id,
                        SecondPerson = relationshipPerson,
                        RelationshipType = relationshipType
                    };

                    await _context.Relationships.AddAsync(relationshipToAdd);
                    await _context.SaveChangesAsync();
                }


                var personRelationshipsInfo = new List<PersonRelationshipInfo>();
                foreach(var personRelationship in _context.Relationships.Where(r => r.FirstPersonId
== id))
                {
                    var personRelationshipInfo = new PersonRelationshipInfo
                    {
                        Type = personRelationship.RelationshipType.Name,
                        Id = personRelationship.Id
                    };

                    if(personRelationship.FirstPersonId == id)
                    {
                        personRelationshipInfo.Person = _mapper.Map<PersonInfoResponse>(
                            personRelationship.SecondPerson);
                    }
                    else
                    {
                        personRelationshipInfo.Person = _mapper.Map<PersonInfoResponse>(
                            personRelationship.FirstPerson);
                    }

                    personRelationshipsInfo.Add(personRelationshipInfo);
```

```
                }
                return new JsonResult(personRelationshipsInfo);
            }
            catch(Exception ex)
            {
                Log.Error(ex, "Unexpected exception during request");
            }

            var dbInternalError = new JsonResult("Помилка бази даних")
            {
                StatusCode = 500
            };
            return dbInternalError;
        }

        [HttpPut, ActionName("{id}")]
        public async Task<IActionResult> UpdatePersonByIdAsync([FromRoute] uint id, [FromBody]
PersonUpdateRequest personUpdateRequest)
        {
            try
            {
                var person = await _context.Persons
                    .Include(p => p.PersonDocument)
                    .FirstOrDefaultAsync(p => p.Id == id);
                if(person == null)
                {
                    Log.Error("Person was not found. PerosnId = {0}", id);

                    var notFoundResult = new JsonResult("Дану особу не було знайдено")
                    {
                        StatusCode = 404
                    };
                    return notFoundResult;
                }

                var personNames = personUpdateRequest.FullName.Split(' ');
                if(personNames.Length != 3)
                {
                    Log.Error("Fullname is invalid. Fullname = {0}",
                        personUpdateRequest.FullName);

                    var badRequestResult = new JsonResult("Невірно введено ПІБ")
                    {
                        StatusCode = 400
                    };
                    return badRequestResult;
                }

                if (personUpdateRequest.SocialSecurityId != null)
                {
                    var socialSecurity = await _context.SocialSecurities
                        .FirstOrDefaultAsync(s => s.Id == personUpdateRequest.SocialSecurityId);
                    if(socialSecurity == null)
                    {
                        Log.Error("Social Security is not found. Social security Id = {0}",
                            personUpdateRequest.SocialSecurityId);

                        var notFoundResult = new JsonResult("Неможливо знайти даний рівень
соціального забезпечення")
                        {
                            StatusCode = 404
                        };
                        return notFoundResult;
                    }
                }

                _mapper.Map(personUpdateRequest, person);

                person.LastName = personNames[0];
                person.FirstName = personNames[1];
                person.MiddleName = personNames[2];

                if(personUpdateRequest.DocumentTypeId != null &&
                    personUpdateRequest.IssuingAuthority != null &&
                    personUpdateRequest.DocumentNumber != null &&
                    personUpdateRequest.DocumentIssueDate != null &&
                    !string.IsNullOrEmpty(personUpdateRequest.DocumentSeries))
```

```
                {
                    var personDocumentType = await _context.PersonDocumentTypes
                        .FirstOrDefaultAsync(s => s.Id == personUpdateRequest.DocumentTypeId);
                    if(personDocumentType == null)
                    {
                        Log.Error("Person document type is not found. Person document type Id =
{0}",
                            personUpdateRequest.SocialSecurityId);

                        var notFoundResult = new JsonResult("Неможливо знайти даний тип документу
особи")
                        {
                            StatusCode = 404
                        };
                        return notFoundResult;
                    }

                    if(person.PersonDocument != null)
                    {
                        person.PersonDocument.PersonDocumentTypeId =
(uint)personUpdateRequest.DocumentTypeId;
                        person.PersonDocument.Number = (uint)personUpdateRequest.DocumentNumber;
                        person.PersonDocument.IssuingAuthority =
(uint)personUpdateRequest.IssuingAuthority;
                        person.PersonDocument.Series = personUpdateRequest.DocumentSeries;
                        person.PersonDocument.IssueDate =
(DateTime)personUpdateRequest.DocumentIssueDate;
                    }
                    else
                    {
                        person.PersonDocument = new PersonDocument
                        {
                            PersonDocumentTypeId = (uint)personUpdateRequest.DocumentTypeId,
                            Number = (uint)personUpdateRequest.DocumentNumber,
                            IssuingAuthority = (uint)personUpdateRequest.IssuingAuthority,
                            Series = personUpdateRequest.DocumentSeries
                        };
                    }
                }

                await _context.SaveChangesAsync();

                Log.Information("Person was successfully updated. " +
                    "Person Id = {0}", person.Id);

                return new JsonResult(person.Id);
            }
            catch(Exception ex)
            {
                Log.Error(ex, "Unexpected exception during request");
            }

            var dbInternalError = new JsonResult("Помилка бази даних")
            {
                StatusCode = 500
            };
            return dbInternalError;
        [HttpDelete, ActionName("{id}")]
        public async Task<IActionResult> DeletePersonByIdAsync([FromRoute] uint id)
        {
            try
            {
                var person = await _context.Persons.FirstOrDefaultAsync(p => p.Id == id);

                if(person == null)
                {
                    Log.Error("Person was not found. PerosnId = {0}", id);

                    var notFoundResult = new JsonResult("Дану особу не було знайдено")
                    {
                        StatusCode = 404
                    };
                    return notFoundResult;
                }

               _context.Persons.Remove(person);
                await _context.SaveChangesAsync();
```

```
                return new JsonResult("Особу було успішно видалено");
            }
            catch(Exception ex)
            {
                Log.Error(ex, "Unexpected exception during request");
            }

            var dbInternalError = new JsonResult("Помилка бази даних")
            {
                StatusCode = 500
            };
            return dbInternalError;
        }
```

IncidentsController.cs

```
using AutoMapper;
using DomesticAbuseRegistry.Common.Utils.Interfaces;
using DomesticAbuseRegistry.Data.Contexts;
using DomesticAbuseRegistry.Data.Entities;
using DomesticAbuseRegistry.Models.Incidents;
using DomesticAbuseRegistry.Models.RiskAssessments;
using DomesticAbuseRegistry.Models.Users;
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Identity;
using Microsoft.AspNetCore.Mvc;
using Microsoft.EntityFrameworkCore;
using Serilog;
using System.Linq;
using System.Security.Claims;

namespace DomesticAbuseRegistry.Controllers
{
    [Route("incidents/[action]")]
    [ApiController]
    [Authorize]
    public class IncidentsController : ControllerBase
    {
        private readonly RegistryContext _context;
        private readonly IMapper _mapper;

        public IncidentsController(RegistryContext context, IMapper mapper)
        {
            _context = context;
            _mapper = mapper;
        }
        [HttpGet, ActionName("{id}")]
        public async Task<IActionResult> GetIncidentByIdAsync([FromRoute] uint id)
        {
            try
            {
                var incident = await _context.Incidents
                    .AsSplitQuery()
                    .Include(i => i.IncidentType)
                    .Include(i => i.Qualification)
                    .FirstOrDefaultAsync(i => i.Id == id);

                if(incident == null)
                {
                    Log.Error("No incidents were found");

                    var notFoundResult = new JsonResult("Дану подію не було знайдено")
                    {
                        StatusCode = 404
                    };
                    return notFoundResult;
                }

                var incidentsInfoResponse = _mapper.Map<IncidentDetailedInfoResponse>(incident);
                var incidentPersons = _mapper.Map<List<IncidentPersonInfo>>(
                    await _context.IncidentPersons
                        .AsSplitQuery()
                        .Include(p => p.PersonType)
                        .Include(p => p.Person)
                        .Where(p => p.IncidentId == incidentsInfoResponse.Id)
                        .ToListAsync()
```

```
                    );

                if(incidentPersons == null || !incidentPersons.Any())
                {
                    Log.Error("No persons found for given incident. " +
                        "incidentId = {0}", incidentsInfoResponse.Id);

                    return new JsonResult(incidentsInfoResponse);
                }

                var personsTypeGroups = incidentPersons.GroupBy(pt => pt.PersonType);
                foreach(var personType in personsTypeGroups)
                {
                    incidentsInfoResponse.IncidentPersons.Add(
                        personType.Key, personType.Select(pt => pt.PersonInfo).ToList());
                }

                Log.Information("Incident info was successfully retrieved. " +
                    "IncidentId = {0}", incidentsInfoResponse.Id);

                if(incident.StatusId != 2)
                {
                    incident.StatusId = 2;
                    incident.ReceiptDate = DateTime.UtcNow;

                    await _context.SaveChangesAsync();
                }

                return new JsonResult(incidentsInfoResponse);
            }
            catch(Exception ex)
            {
                Log.Error(ex, "Unexpected exception during request");
            }

            var dbInternalError = new JsonResult("Помилка бази даних")
            {
                StatusCode = 500
            };
            return dbInternalError;
        }
        [HttpPost, ActionName("")]
        public async Task<IActionResult> AddIncidentAsync([FromBody] IncidentAddRequest
addIncidentRequest)
        {
            try
            {
                if(_context.Incidents.Any(i => i.RegistryNumber ==
addIncidentRequest.RegistryNumber))
                {
                    Log.Error("Given incident already exists in Db." +
                        "Incident registry number = {0}", addIncidentRequest.RegistryNumber);

                    var BadRequestResult = new JsonResult("Дана подія вже існує")
                    {
                        StatusCode = 400
                    };
                    return BadRequestResult;
                }

                var addIncidentPersonsTypesIds = addIncidentRequest.IncidentPersons!
                    .Select(ip => ip.Key);

                foreach(var addIncidentPersonsTypeId in addIncidentPersonsTypesIds)
                {
                    if (!_context.PersonTypes.Any(pt => pt.Id == addIncidentPersonsTypeId))
                    {
                        Log.Error("No such incident persons type is found in Db. " +
                            "Incident persons type id = {0}", addIncidentPersonsTypeId);

                        var NotFoundResult = new JsonResult("Даний тип особи у події не було
знайдено")
                        {
                            StatusCode = 404
                        };
                        return NotFoundResult;
                    }
```

```
                }

                var addIncidentPersonsCount = addIncidentRequest.IncidentPersons!.Select(ip =>
ip.Value.Count);
                if(addIncidentPersonsCount.Any(c => c == 0))
                {
                    Log.Error("No persons are given for some group. Incident persons count = [{0}]",
                        string.Join(", ", addIncidentPersonsCount));

                    var BadRequestResult = new JsonResult("Не дано осіб для якоїсь з груп")
                    {
                        StatusCode = 400
                    };
                    return BadRequestResult;
                }

                var addIncidentPersonsIds = addIncidentRequest.IncidentPersons!.SelectMany(ip =>
ip.Value);

                foreach(var addincidentPersonId in addIncidentPersonsIds)
                {
                    if(addIncidentPersonsIds.Count(id => id == addincidentPersonId) > 1)
                    {
                        Log.Error("Given person can't be mentioned more than one time." +
                            "Incident person id = {0}", addincidentPersonId);

                        var BadRequestResult = new JsonResult("Дана особа не може " +
                            "бути обрана двічі")
                        {
                            StatusCode = 400
                        };
                        return BadRequestResult;
                    }

                    if(!_context.Persons.Any(p => addincidentPersonId == p.Id))
                    {
                        Log.Error("No such person is found in Db." +
                            "Incident person id = {0}", addincidentPersonId);

                        var NotFoundResult = new JsonResult("Дана особа не була знайдена")
                        {
                            StatusCode = 404
                        };
                        return NotFoundResult;
                    }
                }

                if(!_context.IncidentTypes.Any(it => it.Id == addIncidentRequest.IncidentTypeId))
                {
                    Log.Error("No such incident type is found in Db." +
                        "Incident type id = {0}", addIncidentRequest.IncidentTypeId);

                    var NotFoundResult = new JsonResult("Даний тип події не було знайдено")
                    {
                        StatusCode = 404
                    };
                    return NotFoundResult;
                }

                if(!_context.Qualifications.Any(it => it.Id == addIncidentRequest.QualificationId))
                {
                    Log.Error("No such qualification is found in Db." +
                        "Qualification id = {0}", addIncidentRequest.QualificationId);

                    var NotFoundResult = new JsonResult("Дану кваліфікацію події не було знайдено")
                    {
                        StatusCode = 404
                    };
                    return NotFoundResult;
                }

                var incident = new Incident();
                _mapper.Map(addIncidentRequest, incident);

                var incidentPersons = new List<IncidentPerson>();
                foreach (var addIncidentPerson in addIncidentRequest.IncidentPersons!)
                {
```

```
                var personsType = await _context.PersonTypes
                        .FirstAsync(pt => pt.Id == addIncidentPerson.Key);
                var persons = await _context.Persons
                    .Where(p => addIncidentPerson.Value.Contains(p.Id))
                    .ToListAsync();
                var incidentPersonsToAdd = persons.Select
                    (p =>
                        new IncidentPerson
                        {
                            PersonId = p.Id,
                            PersonTypeId = personsType.Id
                        }
                    ).ToList();

                incidentPersons.AddRange(incidentPersonsToAdd);
            }

            incident.IncidentPersons = incidentPersons;
            incident.UserId = UInt32.Parse(User.FindFirstValue("userId"));

            await _context.Incidents.AddAsync(incident);
            await _context.SaveChangesAsync();

            Log.Information("Incident was successfully added to Db. Incident Id = {0}",
                incident.Id);

            var riskAssesment = new RiskAssessment
            {
                Incident = incident
            };

            await _context.RiskAssessments.AddAsync(riskAssesment);
            await _context.SaveChangesAsync();

            var questions = await _context.Questions.ToListAsync();
            var questionAnswers = new List<QuestionAnswer>();
            foreach(var question in questions)
            {
                questionAnswers.Add(new QuestionAnswer
                {
                    Question = question,
                    RiskAssessment = riskAssesment
                });
            }

            await _context.QuestionAnswers.AddRangeAsync(questionAnswers);
            await _context.SaveChangesAsync();

            return new JsonResult(new IncidentAddResponse{ Id = incident.Id });
        }
        catch(Exception ex)
        {
            Log.Error(ex, "Unexpected exception during request");
        }

        var dbInternalError = new JsonResult("Помилка бази даних")
        {
            StatusCode = 500
        };
        return dbInternalError;
    }
    [HttpPut, ActionName("{id}")]
    public async Task<IActionResult> UpdateIncidentInfoAsync([FromRoute] uint id, [FromBody]
IncidentUpdateRequest updateIncidentRequest)
    {
        try
        {
            var incident = await _context.Incidents
                .Include(i => i.IncidentPersons)
                .FirstOrDefaultAsync(i => i.Id == id);
            if(incident == null)
            {
                Log.Error("Дану подію не було знайдено in Db. Incident id = {0}", id);

                var NotFoundResult = new JsonResult("Дану подію не було знайдено")
                {
                    StatusCode = 404
```

```
                    };
                    return NotFoundResult;
                }

                if(!_context.IncidentTypes.Any(it => it.Id == updateIncidentRequest.IncidentTypeId))
                {
                    Log.Error("No such incident type is found in Db." +
                        "Incident type id = {0}", updateIncidentRequest.IncidentTypeId);

                    var NotFoundResult = new JsonResult("Даний тип події не було знайдено")
                    {
                        StatusCode = 404
                    };
                    return NotFoundResult;
                }

                if(!_context.Qualifications.Any(it => it.Id ==
updateIncidentRequest.QualificationId))
                {
                    Log.Error("No such qualification is found in Db." +
                        "Qualification id = {0}", updateIncidentRequest.QualificationId);

                    var NotFoundResult = new JsonResult("Дану кваліфікацію події не було знайдено")
                    {
                        StatusCode = 404
                    };
                    return NotFoundResult;
                }

                if(updateIncidentRequest.IncidentPersons != null &&
updateIncidentRequest.IncidentPersons.Count > 0)
                {
                    var addIncidentPersonsTypesIds = updateIncidentRequest.IncidentPersons!
                        .Select(ip => ip.Key);

                    foreach(var addIncidentPersonsTypeId in addIncidentPersonsTypesIds)
                    {
                        if (!_context.PersonTypes.Any(pt => pt.Id == addIncidentPersonsTypeId))
                        {
                            Log.Error("No such incident persons type is found in Db. " +
                                "Incident persons type id = {0}", addIncidentPersonsTypeId);

                            var NotFoundResult = new JsonResult("Даний тип особи у події не було
знайдено")
                            {
                                StatusCode = 404
                            };
                            return NotFoundResult;
                        }
                    }

                    var addIncidentPersonsIds = updateIncidentRequest.IncidentPersons!.SelectMany(ip
=> ip.Value);

                    foreach(var addincidentPersonId in addIncidentPersonsIds)
                    {
                        if(addIncidentPersonsIds.Count(id => id == addincidentPersonId) > 1)
                        {
                            Log.Error("Given person can't be mentioned more than one time." +
                                "Incident person id = {0}", addincidentPersonId);

                            var BadRequestResult = new JsonResult("Дана особа не може" +
                                "бути обрана двічі")
                            {
                                StatusCode = 400
                            };
                            return BadRequestResult;
                        }

                        if(!_context.Persons.Any(p => addincidentPersonId == p.Id))
                        {
                            Log.Error("No such person is found in Db." +
                                "Incident person id = {0}", addincidentPersonId);

                            var NotFoundResult = new JsonResult("Дана особа не була знайдена")
                            {
                                StatusCode = 404
```

```
                        };
                        return NotFoundResult;
                    }
                }

                var incidentPersons = new List<IncidentPerson>();
                foreach (var addIncidentPerson in updateIncidentRequest.IncidentPersons!)
                {
                    var personsType = await _context.PersonTypes
                            .FirstAsync(pt => pt.Id == addIncidentPerson.Key);
                    var persons = await _context.Persons
                        .Where(p => addIncidentPerson.Value.Contains(p.Id))
                        .ToListAsync();
                    var incidentPersonsToAdd = persons.Select
                        (p =>
                            new IncidentPerson
                            {
                                PersonId = p.Id,
                                PersonTypeId = personsType.Id
                            }
                        ).ToList();

                    incidentPersons.AddRange(incidentPersonsToAdd);
                }

                incident.IncidentPersons.ToList().AddRange(incidentPersons);
            }

            _mapper.Map(updateIncidentRequest, incident);

            if(!string.IsNullOrEmpty(updateIncidentRequest.TakenActions) &&
                incident.StatusId != 3)
            {
                incident.StatusId = 3;
                incident.ExecutionDate = DateTime.UtcNow;
            }

            await _context.SaveChangesAsync();

            Log.Information("Incident was successfully updated. Incident Id = {0}",
                incident.Id);

            return new JsonResult(new IncidentUpdateResponse { Id = incident.Id });
        }
        catch(Exception ex)
        {
            Log.Error(ex, "Unexpected exception during request");
        }

        var dbInternalError = new JsonResult("Помилка бази даних")
        {
            StatusCode = 500
        };
        return dbInternalError;
    }
}
```

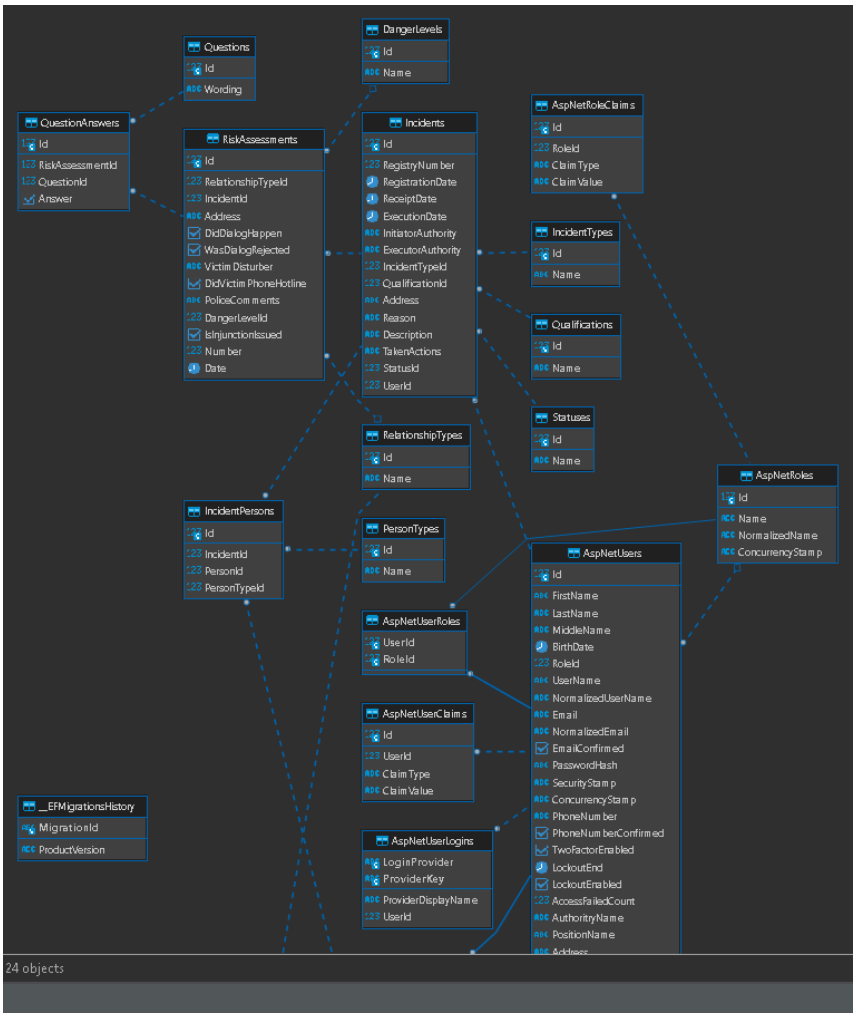
ІЛЮСТРАТИВНА ЧАСТИНА

ЗАХИЩЕНА СИСТЕМА ЗБИРАННЯ ТА АНАЛІЗУВАННЯ ДАНИХ ДЛЯ СПЕЦІАЛЬНИХ ЗАДАЧ. ЧАСТИНА 2. ПІДСИСТЕМА ЗАХИСТУ

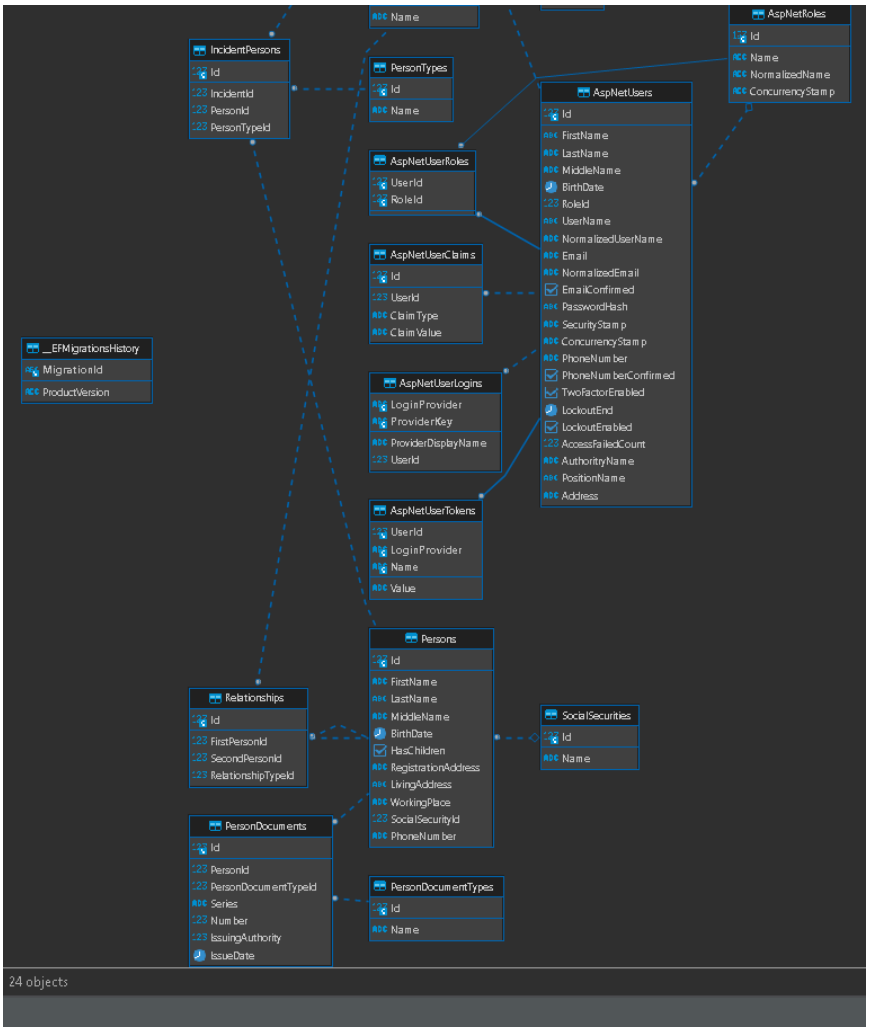
Порівняльна таблиця застосунків для пошуку атак

Комплекси/критерії	Аналіз трафіку	Відповідність стандартам	Ціна	Захист	Простота роботи	Сумарна балів (0-50)
AirMagnet Enterprise	7	8	5	8	5	33
Waidps	4	0	6	4	7	21
Nzyme	3	0	7	3	8	22
Avast Free	2	0	8	2	8	20

Узагальнена структура бази даних



Узагальнена структура бази даних



Результат тестування програмного засобу

200

Response body


```
{
  "incidentType": "Test",
  "qualification": "Test_Qualification",
  "address": "Тестова_адреса",
  "reason": "Тестова_причина",
  "description": null,
  "takenActions": null,
  "id": 24,
  "registryNumber": 12446,
  "registrationDate": "2022-12-18T17:12:32.774Z",
  "receiptDate": null,
  "executionDate": null,
  "initiatorAuthority": "Тестовий_ініціатор",
  "executorAuthority": "Тестовий_виконавець",
  "status": null,
  "incidentPersons": {
    "Постраждалий": [
      {
        "id": 2,
        "fullName": "Testovich Test Testov",
        "documentNumber": 0,
        "documentSeries": null
      }
    ]
  }
}
```


Download

200

Response body


```
[
  {
    "id": 16,
    "type": "Test_Relationship",
    "person": {
      "id": 1,
      "fullName": "Chupryna Ivan Adriyovich",
      "documentNumber": 0,
      "documentSeries": null
    }
  }
]
```


Download

Вікно для додавання token у «Authorization» header

Результат тестування програмного засобу

200

Response body


```
{
  "documentType": "Паспорт",
  "documentIssuingAuthority": 1123,
  "documentIssueDate": "2023-12-18T16:28:28.135Z",
  "birthDate": "2022-12-18T16:28:28.135Z",
  "phoneNumber": "123456789",
  "hasChildren": false,
  "registrationAddress": "Тестова Адреса",
  "livingAddress": "Тестова Адреса",
  "workingPlace": "Тестова робота",
  "socialSecurity": "Test_SocialSecurity",
  "relationships": [
    {
      "id": 16,
      "type": "Test_Relationship",
      "person": {
        "id": 1,
        "fullName": "Chupryna Ivan Adriyovich",
        "documentNumber": 0,
        "documentSeries": null
      }
    }
  ],
  "id": 0,
  "fullName": "Нульовий Тестовий Чоловік",
  "documentNumber": 1234456,
  "documentSeries": "Серія Паспорта"
}
```


Download

Request URL

```
https://localhost:7282/persons
```

Server response

Code	Details
401 *undocumented*	Error: response status is 401

Response headers

```
access-control-allow-origin: *
content-length: 0
date: Sun,18 Dec 2022 16:28:28 GMT
server: Kestrel
www-authenticate: Bearer
```