Міністерство освіти і науки України
Харківський національний університет імені В.Н. Каразіна
Факультет комп'ютерних наук
Спеціальність 125 «Кібербезпека»
Освітня програма «Безпека інформаційних та комунікаційних систем»

«Допущено до захисту»
В.о. зав. кафедрою БІСТ
Ольга МЕЛКОЗЬОРОВА
«__»_____2023 р.

Пояснювальна записка

до кваліфікаційної роботи магістра

на тему: «Аналіз вразливостей та захист баз даних від SQL-ін'єкцій та DoS атак»

оцінка «_____»
Голова ЕК
Олександр ЛЕМЕШКО _____

Керівник: д.т.н., проф. Єсін В.І.
Рецензент: проф.
Виконавець: студентка групи КБ-61
Довгаль Анастасія Костянтинівна

Харків - 2023

РЕФЕРАТ

Пояснювальна записка: 67 с., 37 рисунків, 7 таблиць, 18 джерел.

У роботі проведено аналіз актуальних вразливостей баз даних та програмна реалізація способів їх захиту від SQL-ін‘єкцій та DoS атак.

Метою дипломної роботи є дослідження вразливостей баз даних для виявлення критичних елементів безпеки та реалізації захисту бази даних від SQL-ін‘єкцій та DoS атак.

При виконанні роботи було досліджено характеристики основних типів баз даних та вразливості, характерні для них. Визначено вразливості на рівні проектування, реалізації та конфігурації баз даних та методи захисту баз даних від виявлених вразливостей. Результатом роботи є: відповідні рекомендації щодо забезпечення захисту баз даних, включаючи захист від SQL-ін'єкцій та DoS атак, розроблена в якості прототипу база даних автоматизованої системи автозаправної станції та відповідне ПЗ для роботи з нею, які пройшли відповідне тестування на предмет дієвості засобів захисту. Розробка бази даних здійснювалася засобами системи керування базами даних MySQL, а розробка інтерфейсу взаємодії з нею – мовами HTML, CSS, PHP та JavaScript. Логіка засобів захисту бази даних реалізована за допомогою мови програмування PHP без використання сторонніх бібліотек та фреймворків.

Ключові слова: БАЗА ДАНИХ, SQL-ІН'ЄКЦІЯ, АТАКИ DOS, КІБЕРБЕЗПЕКА, ВРАЗЛИВОСТІ, ЗАХИСТ ДАНИХ, ІНФОРМАЦІЙНІ СИСТЕМИ.

ABSTRACT

Explanatory note: 67 p., 37 figures, 7 tables, 18 sources.

The work analyzes the current vulnerabilities of databases and software implementation of ways to prevent them from SQL injections and DoS attacks.

The aim of the thesis is to study the vulnerabilities of databases to identify critical security elements and implement database protection against SQL injections and DoS attacks.

During the performance of the work, the characteristics of the main types of databases and the vulnerabilities characteristic of them were investigated. Vulnerabilities at the level of design, implementation and configuration of databases and methods of protecting databases from identified vulnerabilities are identified. The result of the work is determined on the basis of the conducted vulnerability analysis and software-implemented means of protecting the database against SQL injections and DoS attacks. The development of the database was carried out using the tools of the MySQL database management system, and the development of the interaction interface with it - in the languages HTML, CSS, PHP and JavaScript. The logic of database protection tools is implemented using the PHP programming language without the use of third-party libraries and frameworks.

Keywords: DATABASE, SQL INJECTION, DOS ATTACKS, CYBER SECURITY, VULNERABILITIES, DATA PROTECTION, INFORMATION SYSTEMS.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАЧЕНЬ, ТЕРМІНІВ 6
ВСТУП 7
1 ТЕРМІНОЛОГІЧНИЙ БАЗИС ПРЕДМЕТНОЇ ОБЛАСТІ, ЩО РОЗГЛЯДАЄТЬСЯ 9
1.1 Ключові поняття баз даних 9
1.2 Основні мови запитів , що використовуються у різних типах БД 10
1.3 Характеристика основних типів баз даних 14
1.4 Висновки до розділу 22
2 АНАЛІЗ ВРАЗЛИВОСТЕЙ БАЗ ДАНИХ ТА ЗАСОБИ ЇХ ЗАХИСТУ 23
2.1 Визначення вразливостей баз даних 23
2.2 Аналіз вразливостей бази даних 23
2.2.1 Вразливості на рівні проектування бази даних 25
2.2.2 Вразливості на рівні реалізації бази даних 27
2.2.3 Вразливості на рівні конфігурації бази даних 29
2.2.4 Захист даних у базі даних на рівні взаємодії з користувачем 32
2.2.5 Захист даних на рівні взаємодії з базою даних 33
2.2.6 Захист даних у базі даних на рівні мережі 35
2.3 Характеристика input-ін'єкцій та способи захисту від них 36
2.4 Характеристика DoS-атак та способи захисту від них 41
2.5 Висновки до розділу 46
3 ПРОЕКТУВАННЯ ТА ПРАКТИЧНА РЕАЛІЗАЦІЯ ЗАСОБІВ ЗАХИСТУ БАЗИ ДАНИХ АВТОМАТИЗОВАНОЇ СИСТЕМИ АВТОЗАПРАВНОЇ СТАНЦІЇ 47
3.1 Визначення вимог до безпеки бази даних, що проектується 47
3.2 Проектування структури застосунку для взаємодії з базою даних 48
3.3 Проектування бази даних 51
3.4 Реалізація базових засобів безпеки бази даних 54
3.6 Реалізація захисту від SQL-ін'єкцій 58

3.7 Реалізація захисту від атак типу DoS 62
3.8 Тестування реалізованих засобів безпеки 64
3.9 Висновок до розділу 68
ВИСНОВОК 69
ПЕРЕЛІК ДЖЕРЕЛ ТА ПОСИЛАННЯ 71
ДОДАТОК А 73
ДОДАТОК Б 79
ДОДАТОК В 93
ДОДАТОК Г 94
ДОДАТОК Д 96
ДОДАТОК Е 98
ДОДАТОК Ж 99
ДОДАТОК З 101

ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАЧЕНЬ, ТЕРМІНІВ

БД	База Даних
СКБД	Система Керування Базами Даних
SQL	Structured Query Language
ОС	Операційна Система
DoS	Denial Of Service attack
XML	eXtensible Markup Language
JSON	JavaScript Object Notation
JSP	Java Server Page technology
UX	User experience
RDBMS	Реляційні бази дани
DDB	Distributed Database

ВСТУП

Всесвітня мережа та інформаційні технології відіграють ключову роль у сучасному світі, надаючи можливість зберігання, обробки та обміну даними. Однак, разом з цими можливостями, зростає загроза для інформаційної безпеки. Цільовими об'єктами для потенційних атак є бази даних, що містять важливу інформацію, таку як особисті дані, конфіденційні дані, комерційні та господарські документи.

Зокрема, дві з найпоширеніших та найнебезпечніших атаки на бази даних - це SQL-ін'єкції та атаки з використанням відмови в обслуговуванні (DoS). SQL-ін'єкції дозволяють зловмисникам впроваджувати шкідливі SQL-запити в інформаційну систему, використовуючи недоліки в програмному забезпеченні. У разі успішного виконання, ці атаки можуть призвести до витоку, втрати чи деформації даних. Атаки DoS спрямовані на переповнення ресурсів інформаційної системи або мережі, що може призвести до її відмови та недоступності для легітимних користувачів.

Дослідження передбачає аналіз потенційних вразливостей, властивих базам даних, типів атак зловмисника з метою обґрунтованого впровадження відповідних заходів безпеки.

Основними завданнями дослідження є:

1) Аналіз вразливостей баз даних – визначення слабких місць, які можуть стати точкою входу для зловмисників.

2) Аналіз типів атак з метою розуміння їх сутності та механізмів дії.

3) Побудова бази даних автоматизованої системи автозаправної станції.

4) Аналіз механізмів, що забезпечують безпеку баз даних.

5) Розробка практичних рекомендацій щодо використання різних механізмів, що забезпечують захист бази даних.

6) Розробка механізмів, які забезпечують безпеку спроектованої бази даних.

7) Тестування реалізованих засобів захисту

Об'єктом дослідження є процес створення захищених баз даних в умовах впливу різних типів атак та наявних вразливостей в існуючому програмному забезпеченні баз даних та СКБД.

Предметом дослідження є методи та засоби захисту баз даних від визначених вразливостей.

Метою дипломної роботи є аналіз актуальних вразливостей, властивих базам даних, для побудови ефективної системи їх захисту, особливо від SQL-ін'єкцій та DoS атак.

Актуальність роботи обумовлена тим, що кількість кібератак на конфіденційні дані користувачів у базах даних має тенденцію зростати з кожним роком, що призводить до значних фінансових та репутаційних втрат компаній. Розроблені методи і рекомендації можуть бути використані організаціями для підвищення рівня кібербезпеки та захисту користувачів від потенційних загроз.

1 ТЕРМІНОЛОГІЧНИЙ БАЗИС ПРЕДМЕТНОЇ ОБЛАСТІ, ЩО РОЗГЛЯДАЄТЬСЯ

1.1 Ключові поняття баз даних

Бази даних є важливою складовою для зберігання та оптимізації великого обсягу інформації. Для ефективного використання таких систем та для гарантії їхньої надійності та зручності використання, необхідно відмінно розуміти ключові поняття, які складають основу цих структур. Наведемо ключові поняття у сфері баз даних:

- База даних (БД): База даних - це організована колекція даних, яка зберігається та підтримується для подальшого доступу та опрацювання. Вона може включати інформацію про різні сутності, такі як клієнти, продукти, замовлення тощо.
- Сутність (Entity): Сутність - це об'єкт або явище в реальному світі, який може бути ідентифікований та описаний з точки зору бази даних. Наприклад, в базі даних для інтернет-магазину сутностями можуть бути клієнти, продукти, замовлення тощо.
- Атрибут (Attribute): Атрибут - це характеристика або властивість сутності, яка описує певний аспект цієї сутності. Наприклад, у сутності "клієнт" атрибутами можуть бути ім'я, адреса, телефон тощо.
- Кортеж (Tuple): Кортеж - це конкретний запис або рядок в таблиці бази даних. Він представляє собою комбінацію значень атрибутів для певної сутності.
- Таблиця (Table): Таблиця - це структурована форма для організації даних в базі даних. Кожна таблиця містить дані про певну сутність і має стовпці, які відповідають атрибутам цієї сутності.
- Ключ (Key): Ключ - це атрибут або комбінація атрибутів, за якими можна ідентифікувати унікальний запис в таблиці. Ключі використовуються для швидкого доступу до даних та для встановлення зв'язків між таблицями.
- Нормалізація (Normalization): Нормалізація - це процес організації даних в базі даних таким чином, щоб уникнути зайвої реплікації та забезпечити

ефективний доступ до інформації. Вона включає розбиття таблиць на менші і більш структуровані частини.

Ці ключові поняття є основою для розуміння та роботи з базами даних. Вони допомагають впорядковувати та організовувати інформацію, забезпечуючи її доступність та цілісність. Поглиблене знання цих понять є важливим для розробки та управління базами даних в будь-якій галузі інформаційних технологій [5].

1.2 Основні мови запитів , що використовуються у різних типах БД

Мови запитів грають важливу роль у взаємодії з базами даних. Вони дозволяють вибирати, оновлювати, видаляти та перетворювати дані, що зберігаються в базі даних. Різні типи баз даних використовують різні мови запитів, або ж мають свої варіанти мови запитів. Розглянемо основні мови запитів та їх застосування в різних типах баз даних.

SQL (Structured Query Language):

SQL є найпоширенішою та загально прийнятою мовою запитів для реляційних баз даних (RDBMS), яка відзначається своєю стандартизацією та потужністю для роботи з реляційними даними. SQL дозволяє розробникам та адміністраторам баз даних виразно виражати запити до даних у вигляді структурованих команд.

Основними операціями SQL є:

1) SELECT (Вибірка): Операція SELECT використовується для вибірки даних з таблиці. За допомогою цієї операції можна витягти певні рядки або колонки з таблиці або виконати об'єднання таблиць для отримання більш складних результатів.

2) INSERT (Вставка): Операція INSERT дозволяє додавати нові дані до таблиці. Ви вказуєте дані, які потрібно додати, та таблицю, в яку вони повинні бути вставлені.

3) UPDATE (Оновлення): Операція UPDATE використовується для зміни існуючих даних у таблиці. Ви вказуєте, які дані потрібно оновити та нові значення для них.

4) DELETE (Видалення): Операція DELETE дозволяє видаляти дані з таблиці. У запиті необхідно вказати, які саме дані потрібно видалити.

NoSQL Query Languages:

У нереляційних базах даних (NoSQL), мови запитів розрізняються від мов SQL, оскільки NoSQL бази даних мають різні моделі даних та вимагають відповідних запитів для роботи з ними. Кожен тип NoSQL бази даних може використовувати свою власну мову запитів або мати свої особливість в існуючих мовах.

Основні типи NoSQL баз даних та їх мови запитів включають:

1) Документ-орієнтовані бази даних (наприклад, MongoDB): У таких базах даних використовується мова запитів, яка ґрунтується на структурі документів (зазвичай у форматі JSON або BSON).

2) Бази даних типу ключ-значення (наприклад, Redis): У таких базах даних мова запитів базується на використанні ключів для отримання значень. Зазвичай, це прості запити, які стосуються встановлення, отримання та видалення даних за ключами.

3) Колонкові бази даних (наприклад, Apache Cassandra): Для роботи з колоночними базами даних використовується спеціалізована мова запитів, яка дозволяє отримувати дані, збережені у вигляді колонок.

Кожен тип NoSQL бази даних має свою мову запитів або специфікації для роботи з даними відповідно до своєї моделі даних. Це важливо враховувати при роботі з NoSQL базами даних, оскільки розуміння мови запитів та особливостей конкретної бази даних є ключовим для ефективної роботи та взаємодії з даними [6].

Xquery:

XQuery - це мова запитів, призначена для роботи з XML-даними. Вона використовується в базах даних, де дані зберігаються у форматі XML. XQuery дозволяє виражати складні запити до XML-структур, витягати піддерева, фільтрувати дані та виконувати перетворення.

Основні особливості XQuery:

- Синтаксис для навігації по XML: XQuery надає зручний спосіб навігації по ієрархічній структурі XML-документів. У вашому прикладі, for $book in /library/book означає, що XQuery обирає всі елементи <book> з кореня XML-документа, які знаходяться у <library>.
- Фільтрація та умови: За допомогою XQuery можна легко фільтрувати дані. У вашому прикладі, where $book/@genre = "Science Fiction" фільтрує результати таким чином, що обираються лише ті елементи <book>, у яких атрибут genre дорівнює "Science Fiction".

За допомогою XQuery можна виконувати різноманітні завдання, пов'язані з обробкою XML-даних, такі як знаходження конкретних даних, обчислення значень або перетворення даних з одного формату в інший.

XQuery використовується не лише для запитів до XML-даних у базах даних, але і в інших сценаріях, де потрібно обробляти XML-дані, такі як обробка веб-сервісів, перетворення XML-даних, пошук та фільтрація XML-інформації [7].

SPARQL:

SPARQL - це мова запитів, призначена для роботи з даними у вигляді графів, яка використовується у семантичних графових базах даних. Вона спеціалізована для роботи з ресурсами, які пов'язані між собою в семантичних мережах та RDF-даних (Resource Description Framework).

Основні особливості SPARQL:

- Графова модель даних: SPARQL використовує графову модель для виразу даних. Графова модель полягає в тому, що дані представлені у вигляді вузлів (ресурсів) та зв'язків між ними. Ця модель ідеально підходить для виразу семантичних зв'язків між ресурсами.
- Запити для пошуку та аналізу зв'язків: SPARQL дозволяє створювати складні запити для пошуку даних та аналізу зв'язків між ресурсами. Ви можете вказати шаблон графу, який ви шукаєте, та SPARQL знайде всі відповідні відношення та ресурси.

- Підтримка RDF-даних: SPARQL розроблена спеціально для роботи з RDF-даними, які широко використовуються для виразу семантичної інформації в Світі Великих Даних та веб-семантиці.

SPARQL широко використовується у семантичних веб-додатках, розробці онтологій, інтелектуальних системах та деяких виданнях баз даних, де зв'язки між даними є ключовими.

OQL (Object Query Language):

OQL - це мова запитів, яка використовується в об'єктно-орієнтованих базах даних (OODBMS). Основна ідея OQL полягає в тому, щоб дозволити розробникам виразно виражати запити до об'єктів, які зберігаються в базі даних, та отримувати об'єкти відповідно до заданих критеріїв пошуку.

Основні особливості OQL:

- Робота з об'єктами: OQL розуміє, що дані в об'єктно-орієнтованих базах даних представлені як об'єкти, які мають властивості та методи. Таким чином, OQL дозволяє виразити запити до цих об'єктів, а не просто до таблиць та рядків даних.
- Фільтрація та умови: OQL надає можливість встановлення різних умов та фільтрів для вибору об'єктів. Ви можете задавати, які об'єкти повинні відповідати певним критеріям.
- Підтримка об'єктної моделі: OQL розроблена для використання в контексті об'єктної моделі даних, і вона добре взаємодіє з об'єктами, їх відношеннями та іншими об'єктно-орієнтованими концепціями.

OQL широко використовується в розробці програмного забезпечення, де об'єктно-орієнтований підхід до даних є ключовим, такий як в сферах розробки програмного забезпечення з великим обсягом даних або в сфері розробки вбудованих систем, де об'єктно-орієнтований дизайн є домінуючим [8].

CQL (Column Query Language):

CQL - це мова запитів, призначена для колоночних баз даних, таких як Apache Cassandra. Вона використовується для вибірки та обробки даних, які зберігаються у вигляді колонок, що дозволяє ефективно робити запити до великих обсягів даних.

Основні особливості CQL:

- Робота з колонками: В колоночних базах даних дані зберігаються у вигляді колонок, а не рядків. CQL надає засоби для виразного вибору конкретних колонок та їх вмісту.
- Спрощений синтаксис: CQL намагається надати спрощений та зрозумілий синтаксис для виразу запитів. Він більш схожий на стандартні SQL-запити, що дозволяє користувачам знайомим з SQL швидко освоїти CQL.

CQL використовується в розподілених базах даних для роботи з великими обсягами даних, де швидкість та ефективність запитів є критичними. Вона часто використовується в сучасних додатках для зберігання та обробки даних в реальному часі.

Різні мови запитів призначені для роботи з різними типами баз даних. SQL - стандарт для реляційних баз даних. У нереляційних базах даних використовуються мови запитів, спеціалізовані для конкретного типу даних. Наприклад, XQuery для XML, SPARQL для семантичних графів, OQL для об'єктно-орієнтованих баз. CQL - мова для колоночних баз даних, що дозволяє ефективно обробляти великі обсяги даних. Розуміння цих мов допомагає вибирати найкращий інструмент для конкретних завдань у роботі з даними [9].

1.3 Характеристика основних типів баз даних

В сучасному інформаційному суспільстві обробка та управління даними є надзвичайно важливими аспектами, що впливають на розвиток різних галузей діяльності. Багато сфер, починаючи від підприємств та організацій, і закінчуючи науковими дослідженнями і побутовими потребами, вимагають надійних засобів для зберігання та доступу до даних. У цьому контексті бази даних стають важливим інструментом, який спрощує організацію, зберігання та управління інформацією. База даних - це структурована колекція даних, яка зберігається та управляється в електронному вигляді. БД може містити інформацію різного типу, таку як текст, числа, зображення, звуки тощо. Головною метою БД є ефективне зберігання, організація та доступ до даних [2].

Система керування базами даних (СКБД): СКБД - це програмне забезпечення, яке дозволяє створювати, модифікувати та оптимізувати бази даних. СКБД забезпечує можливість виконувати операції з даними, такі як додавання, видалення, редагування та вибірка даних [3].

Існуючі типи баз даних можна розділити на наступні типи:

Реляційні БД (Relational Database Management System). Реляційні БД - це найпоширеніший тип баз даних, де дані зберігаються у вигляді таблиць, що мають структуру з рядками і стовпцями. Кожен рядок таблиці відповідає запису в базі даних, і кожен стовпець представляє атрибут або поле даних.

Бази даних такого типу найчастіше використовують для побудови рішень OLTP (Online Transaction Processing). У таких рішеннях СКБД працює з невеликими за розмірами транзакціями, що йдуть великим потоком і потребують від системи мінімальний час відгуку, а також можливість, за певних умов скасувати будь-які зміни у рамках транзакції. Якщо реалізується система, в рамках якої потрібно зберігати значну кількість сутностей (таблиць), з різними типами зв'язків між ними (один до одного, один до багатьох, багато до багатьох), то реляційні СКБД будуть найкращим рішенням реалізації [1].

Основні характеристики реляційних БД:

Таблична структура: Дані в реляційних базах даних зберігаються у вигляді таблиць, де кожен рядок представляє запис, а кожний стовпчик - поле чи атрибут.

Ключі та відносини: Реляційні бази даних використовують ключі для унікальної ідентифікації записів. Вони також підтримують відносини між таблицями, що дозволяє зберігати та забезпечувати цілісність даних.

Мова структурованих запитів: SQL (Structured Query Language) використовується для взаємодії з реляційними базами даних. Вона надає мову для виконання операцій, таких як вибірка, вставка, оновлення та видалення даних.

Нормалізація: Процес нормалізації дозволяє розділити таблиці так, щоб уникнути аномалій та забезпечити ефективну організацію даних.

Цілісність даних: Реляційні бази даних використовують обмеження цілісності для забезпечення правильності та консистентності даних.

Транзакції: Реляційні системи управління базами даних (СУБД) підтримують концепцію транзакцій, які гарантують атомарність, консистентність, ізоляцію та тривалість (ACID-властивості).

Індексація: Використання індексів дозволяє швидше здійснювати пошук та виконання запитів, покращуючи продуктивність.

Масштабованість та оптимізація: Реляційні бази даних можуть бути оптимізовані для виконання запитів та операцій з великим обсягом даних. Також вони можуть бути масштабовані горизонтально або вертикально.

Безпека: Реляційні бази даних надають можливості для визначення прав доступу, що забезпечує захист від несанкціонованого доступу до даних.

Реляційні бази даних є одними з найпоширеніших та широко використовуваних типів баз даних. Вони застосовуються у різних сферах, від бізнесу та фінансів до освіти та державних установ. В якості прикладів реляційних СКБД можна привести MySQL, PostgreSQL, Oracle Database, Microsoft SQL Server, SQLite тощо. Ці СКБД використовуються для зберігання та управління структурованими даними в різних галузях, включаючи бізнес, науку та веб-розробку.

NoSQL БД - Нереляційні БД це клас баз даних, які відхиляються від традиційних реляційних баз даних (RDBMS) і використовують інші моделі збереження та операцій з даними. Вони призначені для збереження та обробки даних, які не підходять для реляційних баз даних, і відповідають на вимоги сучасних застосувань з великим обсягом даних, розподіленими системами та потребами швидкодії. Нереляційні СКБД призначені для роботи з нереляційними або неструктурованими даними, а також для вирішення конкретних вимог щодо швидкодії, масштабованості та гнучкості, які можуть бути важко досягнути з реляційними БД [1].

Основні характеристики нереляційних БД включають:

Типи даних та моделі: NoSQL бази даних можуть використовувати різні моделі даних, такі як ключ-значення, документи, колонки та графи. Кожен тип орієнтований на конкретні вимоги проекту.

Горизонтальне масштабування: NoSQL бази даних легше масштабуються горизонтально, тобто шляхом додавання нових серверів або вузлів до системи, щоб розширити її продуктивність та витривалість.

Гнучкість схеми: NoSQL бази даних зазвичай використовують гнучкі схеми даних, що дозволяє додавати нові поля без необхідності змінювати всю базу даних.

Швидкість та продуктивність: Завдяки своїй структурі та гнучкій моделі даних, NoSQL бази даних можуть бути більш ефективними при обробці великих обсягів даних та розподіленому зберіганні.

Графові бази даних: Деякі NoSQL бази даних спеціалізуються на зберіганні та обробці графових структур, що робить їх ефективними для задач, пов'язаних із зв'язками та мережами.

Безпека та права доступу: NoSQL бази даних можуть мають обмежені можливості щодо забезпечення безпеки та контролю доступу порівняно з традиційними реляційними базами даних.

Транзакції та ACID: Деякі NoSQL бази даних підтримують транзакції та ACID властивості, але не всі. Це може залежати від конкретної реалізації СУБД.

Застосування: NoSQL бази даних часто використовуються для задач, таких як зберігання та обробка великих обсягів даних, аналіз даних у реальному часі, робота з великими масивами неструктурованих даних, тощо.

NoSQL бази даних стали популярними завдяки своїй гнучкості та здатності ефективно вирішувати завдання, для яких традиційні реляційні бази даних можуть бути менш підходящими. Однак важливо враховувати конкретні вимоги проекту та характеристики кожної конкретної NoSQL бази даних при виборі підходящого рішення.

В якості прикладів нереляційних БД можна привести MongoDB, Redis, Apache Cassandra. Нереляційні БД підходять для великої кількості варіантів застосувань, включаючи роботу з великими обсягами даних, сенсорними даними, інтернет-речами (IoT), аналізом соціальних мереж, а також для вирішення завдань, які вимагають високої швидкодії та масштабованості.

Розподілені БД (Distributed Database, DDB):

Розподілені бази даних (РБД) - це системи управління базами даних, які розподілені на кілька фізичних чи логічних машин. Така архітектура дозволяє зберігати та обробляти дані на різних вузлах мережі. Наведемо основні характеристики та аспекти розподілених баз даних[1].

Географічна розподіленість: Розподілені бази даних можуть бути розташовані на різних фізичних машинах, що може включати сервери у різних місцях або навіть країнах. Це особливо корисно для організацій з глобальним присутністю.

Розподілені дані: Дані розбиваються на фрагменти, які зберігаються на різних вузлах. Це може полегшити роботу з великими обсягами даних та забезпечити більш ефективне використання ресурсів.

Розподілені транзакції: Розподілені системи дозволяють виконувати транзакції, які змінюють дані на різних вузлах. Це може включати коміт транзакцій на різних етапах або використання протоколів координації для забезпечення цілісності даних.

Масштабованість: Розподілені бази даних дозволяють масштабувати систему, додаючи нові вузли або розподіляючи навантаження для підтримки зростання обсягів даних та користувачів.

Висока доступність та надійність: Розподілені системи можуть бути налаштовані для забезпечення високої доступності шляхом резервного копіювання даних та автоматичного відновлення після збоїв.

Забезпечення безпеки: Забезпечення конфіденційності, цілісності та доступності даних залишається важливим завданням, і розподілені бази даних включають механізми для забезпечення цих аспектів безпеки.

Реплікація даних: Розподілені бази даних можуть використовувати механізми реплікації для створення копій даних на різних вузлах. Це не тільки забезпечує вищу доступність, але і полегшує швидкий доступ до даних.

Мобільність даних: Розподілені бази даних можуть бути організовані так, щоб підтримувати мобільність даних, забезпечуючи доступ до даних навіть при зміні фізичного місця користувача.

Гетерогенність: Системи можуть включати різні типи баз даних або навіть різні моделі даних, наприклад, реляційні та нереляційні, що дозволяє використовувати різні типи даних для різних завдань.

Шарування даних: Розподілені бази даних можуть використовувати концепцію шарування даних, де окремі шари даних можуть бути доступні різним рівням користувачів або застосунків.

Розподілені бази даних є важливим елементом для багатьох сучасних систем, де важливі такі аспекти, як географічна розподіленість, великі обсяги даних, висока доступність та масштабованість. Однак їх реалізація вимагає обережного проектування та ефективного управління, оскільки вони можуть призводити до додаткових викликів у сферах координації, безпеки та оптимізації.

Ієрархічні бази даних (ІБД):

Ієрархічні бази даних (ІБД) використовують ієрархічну модель даних для зберігання та організації інформації. Ієрархічна база даних (ІБД) використовує структуру даних у формі дерева, де кожен запис (вузол) має одного батька, крім кореневого запису. Кожен вузол може мати декілька дочірніх вузлів, утворюючи ієрархію. ІБД використовуються для моделювання взаємозв'язків в організаційних структурах, деревах файлової системи та інших випадках, де інформація логічно групується у вигляді ієрархії. Наведемо основні аспекти ієрархічних баз даних:

Структура даних: Дані в ІБД організовані у вигляді деревоподібної структури, де кожен запис представляє собою вузол ієрархії.

Батьківські та дочірні записи: Кожен запис (вузол) має одного батька, крім кореневого запису, який не має батьківського запису. Також кожен запис може мати декілька дочірніх записів.

Ієрархічна адресація: До записів зазвичай здійснюється через ієрархічну адресацію, використовуючи шляхи від кореневого вузла до конкретного запису.

Дерево зв'язків: Записи у ІБД утворюють дерево зв'язків, де кожен вузол має тільки одного батька, але може мати багато дочірніх вузлів.

Використання в бізнесі: ІБД часто використовуються для представлення структур даних, таких як організаційні чи інші ієрархії.

Мова запитів: Для взаємодії з ієрархічними базами даних використовують спеціалізовані мови запитів, які дозволяють вибирати та оновлювати дані в ієрархії.

Обмеження: Одним з основних обмежень ієрархічних баз даних є те, що запис може мати тільки одного батька. Це може ускладнювати моделювання взаємодії об'єктів, які взаємодіють з більше ніж однією групою об'єктів.

Ієрархічні бази даних (ІБД) виглядають ефективними у випадках, коли дані можна логічно організувати у вигляді ієрархії, де об'єкти мають батьківські та дочірні зв'язки. Такий підхід особливо корисний для представлення структур, що відображають взаємозв'язки та ієрархії, такі як організаційні структури чи дерева файлової системи.

Однак ієрархічні бази даних мають свої обмеження, зокрема в області гнучкості та розширюваності, оскільки записи можуть мати лише одного батька. Завдяки змінам у технологіях та потребам сучасних додатків, реляційні та нереляційні бази даних стали більш популярними через свою гнучкість та можливості оптимізації для різноманітних сценаріїв.

На основі інформації представленої інформації [1], наведемо порівняльну інформацію між визначеними типами БД у таблиці 1.1.

Таблиця 1.1 - Порівняльна інформацію між визначеними типами БД

№	Назва	Структура даних	Мова запитів	Плюси	Мінуси
1.	Реляційні	Дані зберігаються у вигляді таблиць з рядками та стовпцями, де кожен стовпець має свій тип даних	Використовується мова SQL (Structured Query Language) для операцій з даними	Структурованість даних, ACID властивості, Підтримка для зв'язків	Нефлексибельність схеми, Висока потреба в ресурсах, Повільність в операціях з'єднання

Продовження таблиці 1.1 - Порівняльна інформацію між визначеними типами БД

2.	Нереляційні	Дані зберігаються у різних форматах, таких як JSON, XML, або бінарні об'єкти	Зазвичай використовуються власні мови запитів або API для доступу до даних	Гнучкість схеми, Висока швидкодія, Масштабованість	Відсутність гарантій ACID, Складність запитів, Відсутність зв'язків
3.	Розподілені	Дані розміщені на різних серверах чи вузлах мережі, спільно працюють для обробки та зберігання даних.	Зазвичай використовуються стандартні мови запитів для конкретної реляційної чи нереляційної БД, але можуть також використовувати мови для розподілених систем.	Здатність масштабувати горизонтально для великих обсягів даних. Висока доступність та стійкість до відмов.	Складніше у керуванні та підтримці. Можливі проблеми з консистентністю даних в розподілених середовищах.
4.	Ієрархічні	Дані організовані у вигляді деревоподібної структури, де кожен запис має одного батька і може мати багато дочірніх записів.	Спеціалізовані мови запитів для вибору та оновлення даних у вигляді ієрархії.	Ефективні для представлення ієрархічних структур. Простота виразу деяких взаємозв'язків.	Обмеженість в гнучкості структури даних. Обмеження на кількість батьківських записів для кожного вузла.

Вибір правильного типу бази даних для конкретного проекту є ключовим завданням, оскільки він визначає структуру, продуктивність та зручність роботи з даними.

Реляційні бази даних є загально визнаними та широко використовуваними для багатьох видів даних і додатків, зокрема для бізнес-застосунків та систем обробки транзакцій. Вони забезпечують цілісність даних та ефективність запитів.

NoSQL бази даних стали популярними для зберігання великих обсягів неструктурованих даних, таких як дані соціальних мереж, веб-сенсорів та журналів подій. Вони відзначаються гнучкістю та швидкістю роботи, але можуть потребувати більшої уваги до забезпечення цілісності даних.

Ієрархічні БД: Ефективні для відображення ієрархічних відносин, але можуть бути обмеженими в певних випадках гнучкості та розширюваності.

Розподілені БД: Використовуються для обробки великих обсягів даних в розподілених середовищах. Забезпечують високу доступність та масштабованість, але вимагають додаткових зусиль для управління консистентністю та підтримки. Підходять для сучасних додатків з великою кількістю користувачів та розподіленими вимогами.

Вибір правильної бази даних залежить від конкретних потреб та вимог проекту, тому важливо докладно аналізувати переваги та недоліки кожного типу бази даних перед ухваленням рішення щодо її впровадження [4].

1.4 Висновки до розділу

У даному розділі було розглянуто термінологічний базис предметної області безпеки баз даних. Наведено основні поняття щодо основних типів баз даних та їх ролі у сучасному світі. Розглянуто ключові поняття баз даних та мови запитів, що використовуються у різних типах баз даних та їх основні операції.

2 АНАЛІЗ ВРАЗЛИВОСТЕЙ БАЗ ДАНИХ ТА ЗАСОБИ ЇХ ЗАХИСТУ

2.1 Визначення вразливостей баз даних

Підсумовуючи довготривалий досвід різних міжнародних дослідницьких груп і компаній у сфері безпеки баз даних [9], можна виділити наступні основні загрози безпеки баз даних, які є актуальними на сучасному етапі. Ці загрози пов'язані з різними джерелами, але в даному випадку основну увагу приділяється антропогенним джерелам загроз, які виникають в результаті дій суб'єктів:

- Надмірні та несанкціоновані привілеї: дана вразливість має місце, коли особа отримує додаткові права, які перевищують необхідність для виконання своїх посадових обов'язків, що створює можливість для можливих зловживань.
- Зловживання законними привілеями: існує ризик використання користувачами своїх легітимних прав доступу в незаконних цілях.
- Ін'єкції вводу: дана вразливість включає в себе спроби введення шкідливого коду або даних в систему, що може призвести до компрометації безпеки бази даних.
- Шкідливе програмне забезпечення (ПЗ) - malware: навіть законні користувачі можуть стати посередниками для зловмисників, які використовують шкідливе програмне забезпечення для отримання доступу до важливих даних.
- Слабкі аудиторські сліди або недостатність заходів з аудиту даних: недостатність контролю та аудиту може призвести до виявлення проблем з безпекою занадто пізно, коли вже сталося порушення.
- Незахищеність носіїв інформації: незабезпечені резервні копії даних можуть стати легкою мішенню для атак та втрати важливої інформації.

2.2 Аналіз вразливостей бази даних

Забезпечення безпеки інформації у сучасних інформаційних системах, особливо коли мова йде про бази даних, є критично важливим завданням. Проаналізуємо статистичні дані, що надають компанії CERT, The MITRE Corporation [4] щодо атак на дані у великих компаніях, підхід до безпеки в яких не

можна вважити недостатнім. Згідно статистиці за останні роки були виявлені різні вразливості баз даних, які варто враховувати з огляду на їхні можливі наслідки. Наведемо декілька прикладів для та проаналізуємо яка саме вразливість привела до них:

1) У 2020 році, атака SQL Injection була зафіксована на базу даних Oracle соціальної мережі TikTok. Зловмисники використали вразливість для отримання доступу до користувацьких облікових записів та витоку конфіденційної інформації.

- Очевидною причиною є SQL Injection (SQLI), що залишається однією з найпоширеніших атак на бази даних. Зловмисники намагаються вставити шкідливий SQL-код в запити до бази даних, щоб витягнути, змінити або видалити дані. Сучасні техніки обходу фільтрів можуть ускладнити виявлення і запобігання SQLI.

2) У червні 2021 року була виявлена атака SQL Injection на базу даних Twitter, яка використовує реляційну систему керування базами даних (RDBMS). Зловмисники скористалися вразливістю у веб-додатку Twitter для внесення зловмисного SQL-коду у запити до бази даних.

- Вразливість що привела до даної проблеми – недостатні права доступу, що можуть призвести до ситуацій, коли користувачі мають більше доступу до бази даних, ніж необхідно. Це створює ризик несанкціонованого доступу та зміни даних. Перевірка і вдосконалення систем контролю доступу є нагальним завданням.

3) У лютому 2020 року GitHub став об'єктом однієї з найбільших в історії атак DDoS. Сайт GitHub став недоступним протягом деякого часу внаслідок значного обсягу трафіку, який був спрямований на нього.

- Вразливість, що призвела до цього це те, що GitHub може бути уразливим до атак через великий обсяг трафіку, який важко фільтрувати, та можливість використання ботнетів для створення великої кількості запитів. Атака DDoS була спрямована на інфраструктуру GitHub, завантажуючи її та забороняючи доступ користувачам навпротязі доволі довгого часу.

4) У жовтні 2016 року сервіс DNS-хостингу Dyn, який обслуговує численні популярні веб-сайти та служби, був скерований атакою DDoS. Це призвело до

тимчасової недоступності багатьох великих веб-сайтів, таких як Twitter, Spotify, Reddit та інші.

- Атака була спрямована на сервіс DNS-хостингу Dyn, переповнюючи його запитами та заважаючи обробці легітимного DNS-трафіку. Уразливості в обробці DNS-запитів та недостатнє розподілення ресурсів могли сприяти успішності атаки.

5) У серпні 2020 року сталася значна атака на інфраструктуру AWS, одного з найбільших хмарних провайдерів у світі. Атака була організована з використанням атаки DOS, спрямованої на інфраструктурні компоненти AWS, такі як DNS-сервери. Внаслідок цього велика кількість веб-сайтів та служб, які спиралися на AWS, були недоступні протягом декількох годин.

- Вразливість, що призвела до даної проблеми – атака типу DoS, атака, під час якої зловмисники намагаються зробити систему недоступною шляхом перенавантаження її ресурсів, таких як мережевий трафік, обчислювальна потужність чи пам'ять.

Ці приклади свідчать про те, що вразливості баз даних залишаються актуальними та можуть мати серйозні наслідки для безпеки та конфіденційності даних. Постійний моніторинг та виправлення цих вразливостей є важливими завданнями для забезпечення безпеки бази даних.

2.2.1 Вразливості на рівні проектування бази даних

Процес проектування баз даних є важливим етапом у створенні і розвитку інформаційних систем. На цьому етапі визначаються структура даних, способи їх організації та забезпечення доступу до них. Основні кроки проектування бази даних включають в себе аналіз вимог, моделювання даних, створення схеми бази даних та визначення прав доступу.

На рівні проектування бази даних важливо враховувати вимоги до безпеки. Це включає в себе визначення, які дані потребують особливого захисту, які ролі мають доступ до цих даних, і які правила повинні бути встановлені для їхнього оброблення. Наприклад, конфіденційна особиста інформація може вимагати більшого рівня захисту, ніж загальнодоступні дані.

На етапі проектування бази даних можуть виникати різні вразливості та ризики. Деякі з найпоширеніших включають недостатню автентифікацію та авторизацію, недостатній контроль доступу, використання несанкціонованих API для доступу до даних, недостатній моніторинг та аудит безпеки. Наведемо найбільш розповсюджені аспекти вразливостей бази даних на рівні проектування та результати їх наслідків у таблиці 2.1 [10].

Таблиця 2.1 - Аспекти вразливостей бази даних на рівні проектування

№	Назва	Вразливість	Наслідки
1	Недостатній контроль доступу	Проектант бази даних не налаштовує правильний контроль доступу до таблиць і даних. Усі користувачі мають однаковий рівень доступу.	Це може призвести до незаконного доступу до конфіденційної інформації. Наприклад, зловмисники можуть отримати доступ до особистих даних користувачів.
2	Несанкціоноване використання API	Проектант бази даних використовує застарілі або небезпечні API для доступу до бази даних, не проводячи перевірку їхньої безпеки.	Зловмисники можуть використовувати ці API для виконання SQL-ін'єкцій або інших атак, що може призвести до втрати даних або порушення цілісності.
3	Недостатній аудит безпеки	Проектант бази даних не налаштовує систему аудиту безпеки для відстеження дій користувачів та змін в базі даних.	Якщо сталася атака або порушення, то немає можливості встановити, що сталося і які дії були виконані зловмисниками.

Продовження таблиці 2.1- Аспекти вразливостей бази даних на рівні проектування

4.	Недостатнє шифрування даних	При проектуванні бази даних не використовуються механізми шифрування для зберігання інформації.	Якщо зловмисники отримають доступ до файлів бази даних, вони можуть легко прочитати чутливі дані, такі як паролі або особисті дані.

Наведена інформація демонструє, як вразливості баз даних на рівні проектування можуть створювати загрози для безпеки даних. Щоб запобігти таким ситуаціям, важливо дотримуватися найкращих практик проектування баз даних та враховувати аспекти безпеки на кожному етапі розробки.

2.2.2 Вразливості на рівні реалізації бази даних

Процес реалізації баз даних - це етап створення та впровадження бази даних після її проектування. Процес реалізації є невід'ємною частиною їх створення та ефективного використання в інформаційних системах. На даному етапі враховуються попередньо визначені структура та функціональні можливості бази даних. Основні етапи реалізації бази даних включають в себе наступне:

- Створення фізичної структури бази даних. Цей етап передбачає фізичне створення таблиць, полів, індексів та відносин між ними на основі попередньо розробленої схеми бази даних. Фізична структура бази даних повинна відповідати її логічній схемі.

- Налаштування прав доступу. На цьому етапі визначаються права доступу користувачів та ролей. Це включає в себе надання або обмеження можливостей для читання, запису, оновлення та видалення даних. Також важливо встановити правила автентифікації та авторизації.

- Захист даних інструментами шифрування. Шифрування використовується для захисту чутливих даних, щоб надійно захистити їх від

несанкціонованого доступу. На цьому етапі налаштовуються механізми шифрування для зберігання та передачі даних в безпеці.

- Тестування та валідація Перед впровадженням бази даних необхідно провести тестування для перевірки її функціональності та безпеки. Це включає в себе тестування на вразливості, а також перевірку на відповідність вимогам.
- Підтримка та адміністрування - після впровадження бази даних виробляють регулярну підтримку, включаючи резервне копіювання даних, моніторинг її роботи та виявлення можливих проблем.

На етапі реалізації бази даних можуть виникати різні вразливості та ризики. Наведемо найбільш розповсюджені аспекти вразливостей бази даних на цьому рівні та результати їх наслідків у таблиці 2.2 [11].

Таблиця 2.2 - Аспекти вразливостей бази даних на рівні реалізації

№	Назва	Вразливість	Наслідки
1	SQL-ін'єкція	Внесення зловмисного SQL-коду у вхідні дані, які потрапляють у базу даних, може призвести до вразливості системи. Зловмисник може отримати несанкціонований доступ до бази даних.	Зловмисники можуть отримати доступ до, змінити або видалити дані з бази даних. Це може призвести до витоку конфіденційної інформації.
2	Використання застарілих програмних рішень	Використання застарілих версій програмного забезпечення бази даних, які мають відомі вразливості.	Зловмисники можуть використовувати ці вразливості для атак і отримання доступу до бази даних. Це може призвести до незаконного доступу та витоку даних.

Продовження таблиці 2.2 - Аспекти вразливостей бази даних на рівні реалізації

3	Загрози внутрішнього доступу	Недостатній контроль доступу для внутрішніх користувачів бази даних.	Внутрішні користувачі можуть наносити шкоду базі даних або незаконно виносити конфіденційну інформацію.
4	Слабкість аутентифікації та авторизації	Використання слабких методів аутентифікації та авторизації дозволяє зловмисникам легко отримати доступ до системи.	Незаконні користувачі можуть використовувати доступ до бази даних для зміни, видалення або крадіжки даних.
5	Недостатній аудит та журналювання подій	Недостатнє аудитування подій не дозволяє відстежувати та аналізувати активність користувачів та подій в системі.	Важливі події та атаки можуть залишатися непоміченими, що ускладнює виявлення порушень безпеки та їхнє розслідування.

Ці приклади демонструють, як вразливості на рівні реалізації можуть призвести до серйозних наслідків, включаючи виток конфіденційної інформації, порушення приватності та незаконний доступ до бази даних. Щоб уникнути цих наслідків, важливо налагоджувати належні заходи безпеки та вживати заходів для запобігання таким вразливостям.

2.2.3 Вразливості на рівні конфігурації бази даних

Процеси проектування та реалізації баз даних є критичними етапами у створенні і підтримці інформаційних систем. На рівні конфігурації бази даних також існують важливі аспекти безпеки, які вимагають уваги.

Численні вразливості на рівні конфігурації можуть бути використані зловмисниками для незаконного доступу до даних, розкриття конфіденційної інформації або завдання шкоди системі.

Наявність вразливостей на рівні конфігурації баз даних може призвести до серйозних наслідків, таких як витік конфіденційних даних, порушення приватності користувачів і погіршення репутації організації. Для запобігання цим вразливостям важливо налагоджувати належні заходи безпеки, включаючи належний контроль доступу, використання шифрування, встановлення систем моніторингу та аудиту безпеки, а також регулярне оновлення та аналіз конфігурації бази даних.

Наведемо найбільш розповсюджені аспекти вразливостей бази даних на цьому рівні та результати їх наслідків у таблиці 2.3 [12].

Таблиця 2.3 - Аспекти вразливостей бази даних на рівні конфігурації

№	Назва	Вразливість	Наслідки
1	Недостатнє оновлення та патчінг	Недостатнє вчасне оновлення бази даних може призвести до використання відомих вразливостей зловмисниками.	Атаки можуть використовувати вже відомі вразливості, які не були виправлені розробниками, для незаконного доступу.
2	Слабкі паролі та незахищені облікові записи	Використання слабких паролів або наявність незахищених облікових записів може стати легким способом для зловмисників отримати доступ до системи.	Незаконні користувачі можуть легко взяти під контроль облікові записи та використовувати їх для зловживання або видалення даних.

Продовження таблиці 2.3 - Аспекти вразливостей бази даних на рівні конфігурації

3	Відкритий доступ до системних файлів та конфігураційних параметрів	Незахищений доступ до системних файлів та конфігураційних параметрів може дозволити зловмисникам змінювати налаштування бази даних та завдавати шкоди її стабільності.	Зміна системних параметрів може призвести до некоректної роботи бази даних та втрати даних.
4	Недостатнє резервне копіювання та відновлення	Недостатнє проведення регулярних резервних копій та тестування процедур відновлення може призвести до втрати даних в разі аварії або атаки.	Втрата даних може призвести до серйозних фінансових втрат та втрати надійності інформаційної системи.
5	Недостатній моніторинг мережевого трафіку	Недостатній контроль за моніторингом мережевого трафіку між клієнтами та сервером бази даних може узагальнити ризик незауваженого перехоплення або недопущеного модифікації даних під час їх передачі.	Зловмисники можуть отримувати доступ до конфіденційної інформації або навіть вносити зміни в дані, передавані між клієнтом і сервером.

Ці аспекти наголошують на важливості правильної конфігурації та підтримки бази даних для забезпечення її безпеки. Загалом, з наведеної інформації можна зробити висновок, що розуміння та вирішення вразливостей баз даних на рівні конфігурації є критично важливим завданням для забезпечення безпеки даних та надійності інформаційних систем.

2.2.4 Захист даних у базі даних на рівні взаємодії з користувачем

Захист даних на рівні взаємодії з користувачем - це важливий аспект загальної інформаційної безпеки, на якому починається бази даних з зовнішніми факторами. На основі аналізу наведених у попередніх розділах загроз, сформулюємо необхідні принципи захисту даних на рівні взаємодії з користувачем та методи їх реалізації (Таблиця 2.4) [13].

Таблиця 2.4 - Принципи захисту даних на рівні застосунку та методи їх реалізації

№	Назва	Принцип	Реалізація
1	Відстеження та аудит дій користувачів	Запис і аналіз подій і дій користувачів для виявлення підозрілої активності.	Ведення журналів подій (аудиту) з деталізацією дій користувачів та адміністраторів, визначення механізмів моніторингу.
2	Захист від зламу паролів	Запобігання зламу паролів та використанням слабких паролів.	Встановлення політик складних паролів, обмеження спроб вводу паролів, використання механізмів двофакторної аутентифікації.

Продовження таблиці 2.4 - Принципи захисту даних на рівні застосунку та методи їх реалізації

3	Захист від атак типу Cross-Site Scripting (XSS)	Запобігання атакам XSS, при яких зловмисники вставляють скрипти у веб-сторінки, які виконуються в браузері користувача.	Екранування вхідних та вихідних даних, використання бібліотек та фреймворків, які автоматично фільтрують потенційно небезпечний контент.
4	Захист від атак на сесії та крадіжки ідентифікаторів сесій	Запобігання атакам, які можуть використовувати викрадені ідентифікатори сесій для несанкціонованого доступу.	Використання безпечних механізмів керування сесіями, шифрування ідентифікаторів сесій, регулярне оновлення сесій.
5	Захист від атак на виток інформації:	Запобігання витоку конфіденційної інформації через недоліки у застосунку.	Захист конфіденційної інформації за допомогою обмеження доступу до конфіденційних даних, контроль за виведенням інформації.

Ці принципи захисту на рівні взаємодії з користувачем сприяють покращенню безпеки програмного забезпечення, а також захисту конфіденційності, цілісності та доступності даних. Вони важливі для побудови надійного та безпечного програмного застосунку.

2.2.5 Захист даних на рівні взаємодії з базою даних

Захист даних на рівні взаємодії з базою даних є наступним кроком реалізації загальної комплексної інформаційної безпеки даних, на якому відбувається

взаємодія інтерфейсу користувача безпосередньо зі сховищем даних. На основі аналізу наведених у попередніх розділах загроз, сформулюємо необхідні принципи захисту даних на рівні взаємодії з базою даних та методи їх реалізації (Таблиця 2.5) [14].

Таблиця 2.5 – Принципи захисту даних на рівні взаємодії з базою даних та методи їх реалізації

№	Назва	Принцип	Реалізація
1	Шифрування даних	Захист конфіденційних даних шляхом їх шифрування.	Використання шифрування для зберігання та передачі даних, особливо конфіденційних ресурсів.
2	Захист від SQL-ін'єкцій	Попередження атак SQL-ін'єкцій шляхом обробки вхідних даних та параметризованих запитів.	Використання параметризованих запитів, валідація вхідних даних, відмова від динамічного створення SQL-запитів.
3	Контроль доступу до даних	Забезпечення доступу тільки для авторизованих користувачів та обмеження їхніх прав доступу до конкретних даних.	Використання рівнів доступу, ролей та прав доступу до таблиць та записів, налаштування автентифікації користувачів.
4	Захист від несанкціонованого доступу до серверів баз даних	Захист серверів баз даних від несанкціонованого фізичного або мережевого доступу.	Фізична безпека приміщення серверів, використання мережевих засобів безпеки.
5	Захист від внутрішнього загрози	Запобігання діям адміністраторів, спрямованим на зловживання даними або функціональністю бази даних.	Моніторинг активності адміністраторів бази даних, обмеження прав доступу, аудит дій.
6	Захист від витоку інформації через аплікаційні помилки	Запобігання витоку конфіденційної інформації через помилки у програмному коді або неправильну конфігурацію бази даних.	Ретельна перевірка програмного коду на наявність помилок та вразливостей, обмеження доступу до конфіденційних даних.

Наведені принципи захисту даних на рівні бази даних є важливими для забезпечення безпеки і цілісності даних в базі даних, а також для запобігання несанкціонованому доступу та витоку інформації. Реалізація цих принципів вимагає комплексного підходу та систематичного моніторингу безпеки бази даних.

2.2.6 Захист даних у базі даних на рівні мережі

Захист даних на рівні мережі - це важлива складова інформаційної безпеки, оскільки мережа є основним засобом передачі даних між комп'ютерами та серверами. У реалізації комплексного захисту безпеки даних застосунку є наступним кроком реалізації, на якому відбувається реалізація безпеки даних безпосередньо в мережі. На основі аналізу наведених у попередніх розділах загроз, сформулюємо необхідні принципи захисту даних на рівні мережі та методи їх реалізації (Таблиця 2.6) [15].

Таблиця 2.6 – Принципи захисту даних на рівні мережі та методи їх реалізації

№	Назва	Принцип	Реалізація
1	Захист мережевих пристроїв	Забезпечення безпеки мережевих комутаторів, маршрутизаторів, брандмауерів та інших мережевих пристроїв.	Регулярне оновлення ПЗ, встановлення паролів та іншої автентифікації на доступ до пристроїв, відслідковування та блокування незвичайної мережевої активності.
2	Захист від DDoS-атак	Захист мережі та серверів від розподіленого атак DDoS, які можуть спричинити перевантаження та відмову в обслуговуванні.	Використання систем фільтрації та розподілення трафіку, моніторинг мережевої активності для виявлення атак та їх блокування.
3	Сегментація мережі:	Розділення мережі на окремі сегменти для обмеження доступу до даних та ресурсів.	Використання відокремлених мережевих сегментів з різними правилами доступу, контроль трафіку між сегментами.

Продовження таблиці 2.6 – Принципи захисту даних на рівні мережі та методи їх реалізації

4	Обробка помилок та відмовостійкість	Захист від атак на переповнення буфера та інших атак, пов'язаних з некоректною обробкою даних.	Валідація та обмеження введених даних, використання безпечних бібліотек та фреймворків, розробка стійкого до помилок коду.
5	Захист від мережевих атак на рівні додатків	Захист від атак, спрямованих на вразливості додатків, які працюють в мережі.	Аудит додаткового програмного коду на наявність вразливостей, використання веб-файерволів та систем обмеження доступу.
6	Захист від мережевих атак і вразливостей	Захист від різних видів мережевих атак, таких як ARP-отруйні атаки, мережеві вразливості та інші.	Використання систем захисту від мережевих атак, регулярне оновлення мережевого обладнання та програмного забезпечення.

Наведені принципи захисту даних на рівні мережі допомагають забезпечити безпеку та надійність передачі даних в мережі, а також запобігти атакам і витокам інформації. Для забезпечення максимального рівня безпеки на рівні мережі, важливо постійно оновлювати засоби захисту та слідкувати за останніми тенденціями в галузі кібербезпеки. Регулярні аудити безпеки та навчання користувачів стосовно безпеки мережі також є необхідними елементами забезпечення безпеки на рівні мережі.

2.3 Характеристика input-ін'єкцій та способи захисту від них

Input-ін'єкції це вид кібератак, при якому зловмисники використовують некоректно оброблені вхідні дані для виконання небезпечних дій у системі. Ця атака є однією з найпоширеніших та небезпечних загроз для веб-додатків та баз даних. Основною метою input-ін'єкцій є виконання шкідливих команд або запитів до бази даних. Існують різні види ін'єкцій, найбільш поширеними з яких є SQL-ін'єкції, XSS (Cross-Site Scripting) ін'єкції та командні ін'єкції. Кожен вид ін'єкції спрямований на різні шари додатку чи системи.

Зловмисники можуть вводити ін'єкції через різні джерела, такі як поля вводу форм, URL-адреси, HTTP-заголовки і навіть завантажені файли. Тобто, практично будь-яке місце, де програма отримує дані від користувача, може бути потенційним джерелом ін'єкцій.

Розглянемо кожен з видів та наведемо способи захисту:

SQL-ін'єкції. SQL-ін'єкція – це вид атаки на безпеку програмного забезпечення, яка полягає в вставці або виконанні зловмисного SQL-коду в SQL-запитах, які відправляються до бази даних. Ця атака може стати дуже небезпечною, оскільки може призвести до незаконного доступу до даних, втрати конфіденційної інформації, а також пошкодження або видалення даних у базі даних.

Головна причина SQL-ін'єкцій полягає у тому, що застосунки недостатньо перевіряють або екранують вхідні дані, які користувачі надсилають на сервер. Зловмисник може використовувати цю уразливість, вставляючи SQL-код у вхідні дані, що обробляються додатком. Якщо ця атака успішна, SQL-запит створюється з небезпечним SQL-кодом і виконується на сервері бази даних, зазвичай з правами, які має додаток.

Наведемо приклад простої SQL-ін'єкції у базу даних:

Припустимо, є веб-сайт для пошуку продуктів в інтернет-магазині, і користувач може ввести ключове слово для пошуку. SQL-запит для пошуку продукту може виглядати наступним чином:

```sql
SELECT * FROM products WHERE product_name = 'ключове_слово';
```

Рисунок 2.1 – Приклад SQL-запиту для пошуку продукту

Якщо користувач введе запит типу ‘ OR ‘1’=’1 в полі пошуку, то фінальний варіант SQL-запиту прийме наступну форму (Рисунок 2.2):

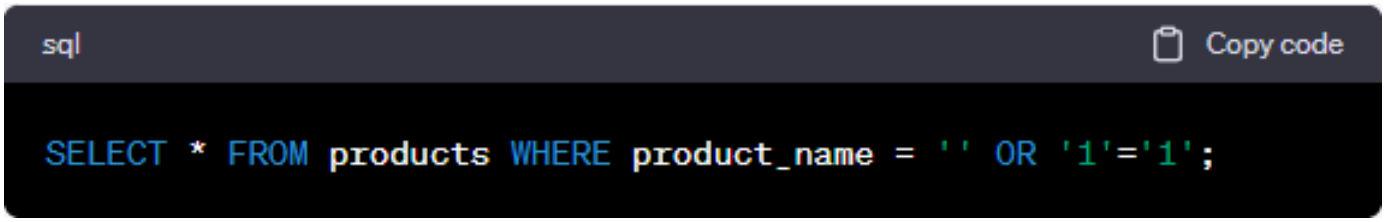


Рисунок 2.2 – Повний вигляд шкідливого запиту

У цьому випадку '1'='1' - завжди істина, тобто цей SQL-запит виведе всі товари в магазині, оскільки умова завжди виконується. Зловмисник може використовувати цей метод для отримання доступу до всієї інформації в базі даних або навіть її редагування, наприклад, додавши до запиту «DELETE * FROM products», зловмисник має можливість знищити інформацію щодо всіх товарів, наявних у базі даних, що у випадку не використання в системі резервних копій може мати критичні наслідки.

Основні наслідки подібних SQL-ін'єкцій включають:

- Незаконний доступ до даних: Зловмисники можуть надолужити конфіденційні дані, включаючи паролі, особисту інформацію користувачів, фінансові дані тощо.
- Маніпуляція даними: Зловмисники можуть змінювати, видаляти або додавати дані до бази даних, що може призвести до втрати інформації або руйнування даних.
- Виконання зловмисного коду: Атаки SQL-ін'єкцією можуть дозволити виконати шкідливий SQL-код на сервері бази даних, що може призвести до небажаних операцій або завантаження шкідливого програмного забезпечення.
- Відмова в обслуговуванні (DoS): Масова атака SQL-ін'єкцією може перевантажити сервер бази даних, що призводить до відмови в обслуговуванні для легальних користувачів.

Для захисту від SQL-ін'єкцій важливо правильно перевіряти та оброблювати вхідні дані, використовувати параметризовані запити, обмежувати права доступу до бази даних і регулярно оновлювати системне програмне забезпечення для

забезпечення безпеки. Також важливо навчати розробників та адміністраторів про перевірку та захист від цього виду атаки.

XSS ін'єкції. XSS ін'єкції (Рисунок 2.4) - це атаки, при яких зловмисники вставляють шкідливі скрипти в веб-сторінки, які потім виконуються на браузерах користувачів. Атаки такого виду зазвичай можуть призвести до наступних наслідків:

- Крадіжка даних користувачів: Зловмисники можуть вкрасти конфіденційні дані, такі як логіни та паролі, з форм вводу на веб-сторінках.
- Сеанс-перехоплення: За допомогою XSS атаки, зловмисники можуть перехоплювати сесійні ідентифікатори користувачів і використовувати їх для несанкціонованого доступу.
- Виведення некоректної інформації: XSS може використовуватися для відображення шкідливого контенту, такого як фальшиві повідомлення про помилки, які можуть обманути користувачів.
- Зловмисне виконання дій від імені користувача: XSS дозволяє зловмисникам виконувати дії під виглядом існуючих користувачів, виконуючі заборонені дії від їх імені.

```
<script>
  fetch("http://зловмисний-сервер.com/вкрадення-сесії")
</script>
```

Рисунок 2.3 – Приклад XSS атаки

Існують три основних типи XSS-ін'єкцій:

- Stored (постійний) XSS: Уразливість поля для введення даних дозволяє зловмисникам зберегти шкідливий скрипт на сервері. Потім цей скрипт відображається на сторінках для всіх користувачів, які їх переглядають.
- Reflected (відображений) XSS: Уразливість передачі даних дозволяє зловмисникам вставити шкідливий скрипт у URL-адресу, який потім відображається на сторінці і виконується під час перегляду користувачами.

- DOM-based (DOM-заснований) XSS: В цьому випадку скрипт не вставляється на сервері, а виконується локально в браузері користувача, змінюючи Document Object Model (DOM) сторінки. Ця форма XSS-ін'єкції складніше виявити та захистити через використання клієнтського JavaScript.

Способи Захисту від XSS ін'єкцій включають наступні методи:

- Валідація та очищення вхідних даних.
- Використання Content Security Policy (CSP) для встановлення політики безпеки щодо завантаження ресурсів на сторінку.
- Екранування виведених даних.
- Валідація та санітизація введених даних на сервері.
- Навчання користувачів про безпеку та ризики виконання скриптів на сторінках.

Загрози від XSS ін'єкцій можуть бути значно зменшені за умови дотримання цих практик та вживання необхідних заходів для захисту від атак.

Командна ін'єкція:

Командна ін'єкція (Рисунок 2.5) - це тип кібератаки, при якій зловмисники вставляють зловмисні команди або командні вирази в вхідні дані, які потім виконуються на сервері або в середовищі виконання команд. Ця атака може відбуватися на серверах або системах, які взаємодіють з командним рядком, таких як веб-сервери, бази даних, операційні системи та інші.

Командна ін'єкція може мати серйозні наслідки, такі як:

- Виконання небажаних команд: Зловмисники можуть виконати небажані системні команди або операції на сервері, що може призвести до втрати конфіденційних даних, зміни конфігурації системи або навіть її злому.
- Здійснення атак від імені користувача: Якщо атака відбувається в контексті користувацької сесії, зловмисники можуть виконувати команди від імені автентифікованих користувачів, що може викликати небажані дії в системі.
- Витік конфіденційних даних: Командна ін'єкція може призвести до витоку конфіденційних даних, таких як паролі, інформація про користувачів та інші конфіденційні дані.

Засоби запобігання командній ін'єкції [16]:

- Валідація та санітизація вхідних даних: Всі вхідні дані, які використовуються в командах або запитах до системи, повинні бути валідовані та санітизовані, щоб вилучити шкідливі символи.
- Використання параметризованих запитів: Замість вставки вхідних даних безпосередньо в команди, слід використовувати параметризовані запити, які розділяють дані від команд та автоматично екранують спеціальні символи.
- Мінімізація використання командного рядка: Обмеження використання командного рядка, особливо у веб-додатках, може значно зменшити ризик командної ін'єкції. Замість цього, використовуйте бібліотеки та функції, які спрощують взаємодію з системою.
- Аудит безпеки: Регулярний аудит безпеки додатків та систем допоможе виявляти можливі уразливості та вчасно їх виправляти.
- Навчання персоналу та розробників: Важливо навчати розробників та адміністраторів систем про небезпеку командних ін'єкцій і правила безпеки взаємодії з командним рядком.

Захист від командної ін'єкції - це критична складова безпеки систем і додатків, і його недоліки можуть призвести до серйозних наслідків.

```
1'; SELECT * FROM customers WHERE '1'='1
```

Рисунок 2.5 – Приклад командної ін’єкції для отримання конфіденційної інформації щодо користувачів у системі

2.4 Характеристика DoS-атак та способи захисту від них

Атака DoS (Відмова в обслуговуванні) є видом кібератаки, в якій зловмисники намагаються призупинити або обмежити доступність послуг, ресурсів або мережевої інфраструктури для законних користувачів, роблячи систему недоступною або надзвичайно повільною. Цей вид атаки вимагає мінімума експлуатації вразливостей

в системі і, зазвичай, ставить своєю метою переповнення ресурсів системи, таких як мережева пропускна здатність, обсяг оперативної пам'яті або потужність обчислення [17].

Основні характеристики атаки DoS включають:

- Перевантаження ресурсів: Зловмисники намагаються надмірно використовувати доступні ресурси, такі як мережеву пропускну здатність, обчислювальну потужність або обсяг пам'яті, з метою переповнення їх і призупинення роботи системи.
- Призупинення послуг: Атака DoS спрямована на призупинення доступності певних послуг або ресурсів, які користувачі зазвичай очікують отримати, таких як веб-сайти, електронна пошта, мережеві додатки тощо.
- Збільшення навантаження: Збільшення навантаження на сервер або мережеву інфраструктуру призводить до повільної роботи та зниження якості обслуговування для легальних користувачів.
- Намагання відмовити в обслуговуванні: Зловмисники намагаються викликати відмову в обслуговуванні, збільшуючи завантаження системи до такого рівня, коли вона не може більше нормально функціонувати.
- Психологічний тиск: Зловмисники можуть намагатися створити психологічний тиск на організацію або індивіда, показуючи їм, що їхні послуги не можуть бути надійними або безпечними.
- Витрати на відновлення: Після атаки DoS організації зазвичай вимушені витрачати велику кількість часу та ресурсів на відновлення систем та послуг, що може призвести до фінансових витрат і втрати репутації.

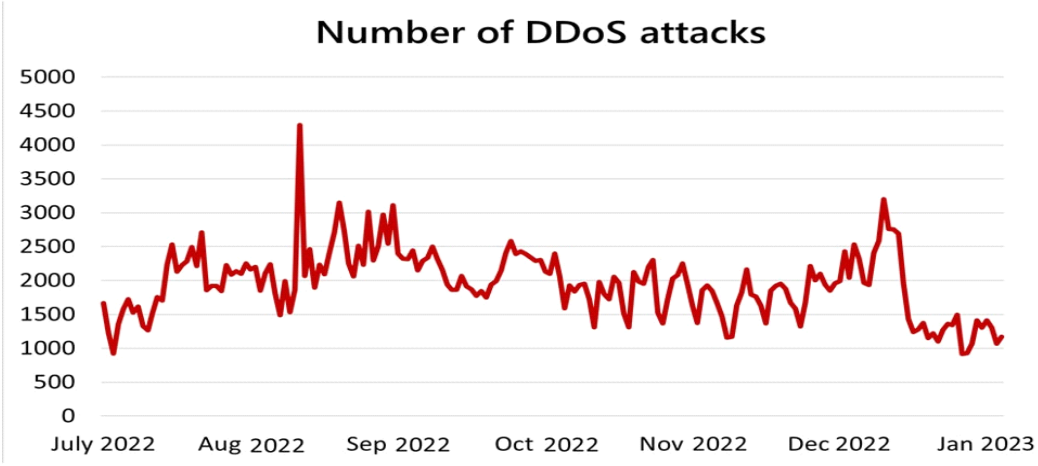


Рисунок 2.6 – Статистика кількості DoS – атак за 2022 рік згідно аналітичним даним Netscout

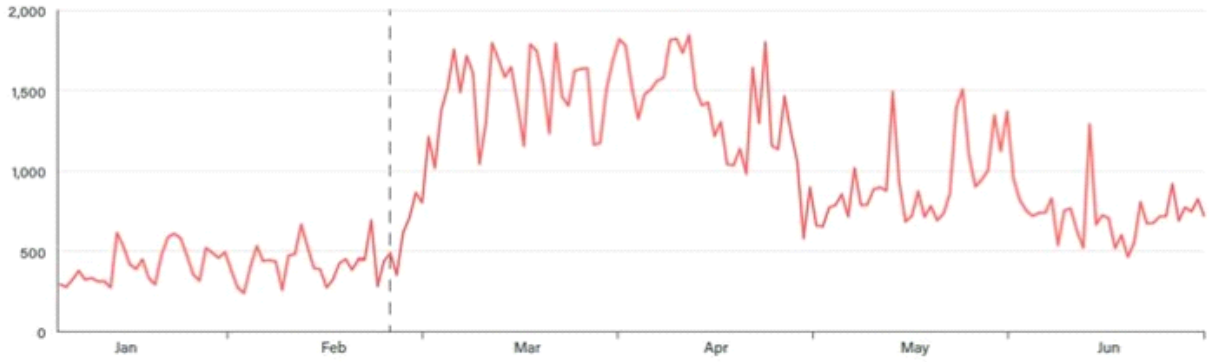


Рисунок 2.7 – Статистика кількості DoS – атак за 2022 рік згідно аналітичним даним Netscout

Атаки DoS (Відмова в обслуговуванні) мають різні варіанти та методи виконання, але всі вони спрямовані на припинення роботи системи або призупинення доступу до послуг. Ось деякі типичні види атак DoS:

- Споживання ресурсів мережі (Network Flooding): У цьому типі атаки зловмисники відправляють велику кількість пакетів даних на цільовий сервер або мережу з метою перевищення їх мережевої пропускної здатності. Це може призвести до того, що законні запити не можуть пройти через перевантажену мережу.

- Атака на протоколи (Protocol Attack): Зловмисники можуть використовувати вразливості у протоколах, таких як TCP/IP, HTTP, або DNS, для призупинення послуг або заволодіння сервером.
- Ампліфікаційні атаки (Amplification Attacks): Ці атаки використовують засоби для збільшення масштабу атаки. Наприклад, атака з використанням UDP-протоколу може відправляти запити до відкритих серверів, що надсилають велику кількість відповідей на адресу цільового сервера, що призводить до перевантаження його мережі
- Атаки на рівні застосунків (Application Layer Attacks): У таких атаках зловмисники спрямовують свої напади на вразливості додатків або веб-сайтів, такі як SQL-ін'єкція, переповнення буфера або відмови в обслуговуванні HTTP (HTTP Flood). Ці атаки можуть вимагати менше ресурсів, але вони можуть бути дуже ефективними.
- Атаки на рівні інфраструктури (Infrastructure Attacks): Атаки можуть бути спрямовані на інфраструктуру, таку як мережеві пристрої, маршрутизатори або DNS-сервери, з метою перевантаження або знищення їх роботи.
- Ресурсомісткі атаки (Resource-Exhaustion Attacks): Зловмисники можуть намагатися вичерпати ресурси сервера, такі як обсяг пам'яті, обчислювальна потужність або відкриті з'єднання, за допомогою інтенсивних запитів або навіть запитів, які вимагають значних обчислень.
- Атаки на рівні DNS (DNS Attacks): Зловмисники можуть використовувати атаки на рівні доменних імен (DNS) для спрямування легітимних запитів на неправильні IP-адреси або призупинення роботи DNS-серверів.
- Атаки на рівні сесій (Session Attacks): Включають в себе атаки на рівні сесій або зламу сесій, з метою призупинення роботи легітимних користувачів або вторгнення в їхні сесії.
- Атаки на рівні SSL/TLS (SSL/TLS Attacks): Ці атаки спрямовані на перевантаження криптографічних пристроїв або обчислювальних ресурсів сервера, які обробляють SSL/TLS-з'єднання.

- Атаки на рівні витрат (Economic Attacks): Включають в себе атаки, які можуть призвести до великих витрат на обробку запитів або передачу даних, такі як атаки на основі мережевого платежу (e.g., атаки на базі мережевих SMS-повідомлень).
- Атаки на рівні флуду (Flood Attacks): Ці атаки включають в себе атаки, в яких зловмисники спрямовують великий обсяг запитів на цільовий ресурс, намагаючись перевантажити його або використати всі доступні ресурси.
- Атаки на рівні мережевого стеку (Network Stack Attacks): Зловмисники можуть використовувати атаки, спрямовані на вразливості в мережевому стеку операційної системи, які можуть призвести до її відмови.
- Атаки на рівні ботнетів (Botnet Attacks): Ботнет - це мережа комп'ютерів, які були заражені шкідливим програмним забезпеченням та можуть бути використані для виконання атаки DoS. Зловмисники можуть керувати ботнетом та відправляти тисячі запитів на цільовий ресурс одночасно.
- Атаки на рівні проксі (Proxy Attacks): Зловмисники можуть використовувати проксі-сервери для маскування своєї ідентичності та розподілу атаки на декілька джерел, щоб ускладнити виявлення та захист.
- Атаки на рівні мережевих пристроїв (Network Device Attacks): Зловмисники можуть атакувати мережеві пристрої, такі як маршрутизатори або комутатори, з метою перекриття або перевантаження мережі.
- Атаки на рівні виробництва (Production Attacks): Ці атаки можуть бути спрямовані на виробничі системи або інфраструктуру, що підтримує продукт або послугу, призводячи до відмови в обслуговуванні та фінансових втрат.

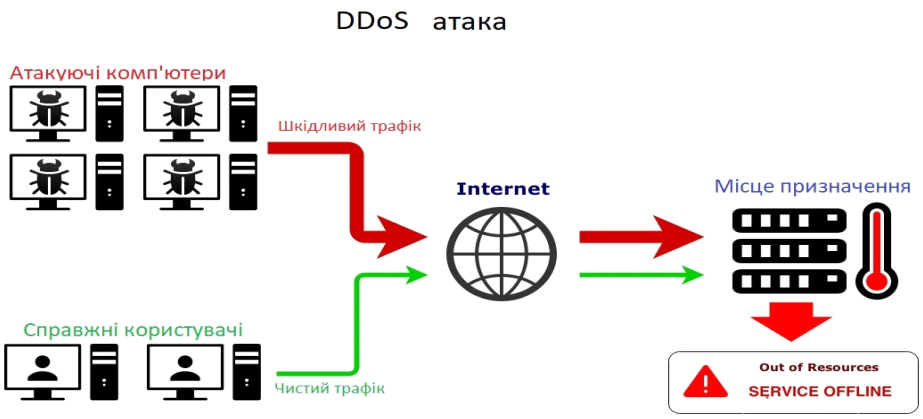


Рисунок 2.8 – Принцип дії атаки типу DoS

Захист від атак DoS включає в себе використання технічних заходів безпеки, таких як моніторинг, розподілення трафіку, регулярне оновлення і захист систем та захисне програмне забезпечення, а також плани відновлення в разі виявлення атаки DoS. Регулярна оцінка ризиків і навчання персоналу щодо реагування на атаки також є важливими аспектами безпеки від атак DoS.

2.5 Висновки до розділу

У даному розділі було проведено аналіз вразливостей баз даних та засоби їх захисту. Визначено основні вразливості бази даних. Проаналізовано та визначено вразливості на різних рівнях, а саме рівні проектування, реалізації та конфігурації. Проведено аналіз основних принципів безпеки баз даних. Сформульовано основні принципи реалізації комплексної безпеки бази даних на рівнях застосунку, бази даних та мережі.

3 ПРОЕКТУВАННЯ ТА ПРАКТИЧНА РЕАЛІЗАЦІЯ ЗАСОБІВ ЗАХИСТУ БАЗИ ДАНИХ АВТОМАТИЗОВАНОЇ СИСТЕМИ АВТОЗАПРАВНОЇ СТАНЦІЇ

3.1 Визначення вимог до безпеки бази даних, що проектується

Перш ніж почати етап проектування та реалізації бази даних та засобів її захисту, необхідно на основі засобів безпеки, що було наведено у пунктах 2.6 – 2.8 зробити висновок яким вимогам безпеки вона повинна відповідати.

Для проектованої бази даних найбільш доречно буде реалізувати комплексний підхід до безпеки та забезпечити засоби її захисту на рівні взаємодії з користувачем, рівні мережі та рівні операцій з даними безпосередньо у базі даних. Для виконання цього завдання необхідно реалізувати захисти від основних вразливостей, що представляють загрозу для бази даних на кожному з цих рівнів, а саме:

- На рівні взаємодії з користувачем забезпечити процеси автентифікації для доступу до даних у базі даних та реалізувати більш складний алгоритм процесу автентифікації користувачів, що мають підвищені права до редагування бази даних.
- На рівні мережі забезпечити засоби захисту від атак типу DoS, що будуть лімітувати кількість запитів до застосунку та бази даних у межах можливостей обробки сервером цих запитів.
- На рівні операцій з даними забезпечити процеси екранування, шифрування та захисту даних під час їх трансферу з клієтської частини застосунку до бази даних та в зворотньому напрямку.

Спираючись на визначені вразливості баз даних сформулюємо наступний список вимог, які повинні бути реалізовані:

- Реалізація процесів авторизації та реєстрації користувачів, які можуть взаємодіяти з базою даних.
- Розмежування прав доступу до різних частин бази даних для різних груп користувачів.

- Реалізація процесів шифрування даних при занесенні їх до бази даних та дешифрування цих даних при відображенні їх користувачу;
- Реалізація засобів захисту від SQL-ін'єкцій.
- Забезпечення багатофакторної автентифікації для користувачів, які мають підвищений рівень доступу до даних.
- Реалізація засобів захисту від атак типу DoS.

3.2 Проектування структури застосунку для взаємодії з базою даних

В якості предметної області для реалізації засобів захисту даних було обрано автоматизовану систему автозаправної станції. Для того, щоб користувач мав можливість взаємодіяти з базою даних, та мати змогу протестувати результати реалізації засобів захисту, необхідно спроектувати та реалізувати застосунок для взаємодії. Застосунок має надавати всі необхідні можливості для взаємодії, мати зручний, інтуїтивний інтерфейс та реалізувати функції обраної предметної галузі. Для того, щоб визначитись зі структурою застосунку, наведемо діаграму основних процесів, що характеризують взаємодію користувача з базою даних (Рисунок 3.1).

Спираючись на визначені процеси, можемо розробити програмний код застосунку (Додаток Б), що реалізує необхідний функціонал, та розробити діаграму взаємодії програмних модулів, що відображує склад інформаційних модулів застосунку, що реалізовано та зв’язки між ними (Додаток В).

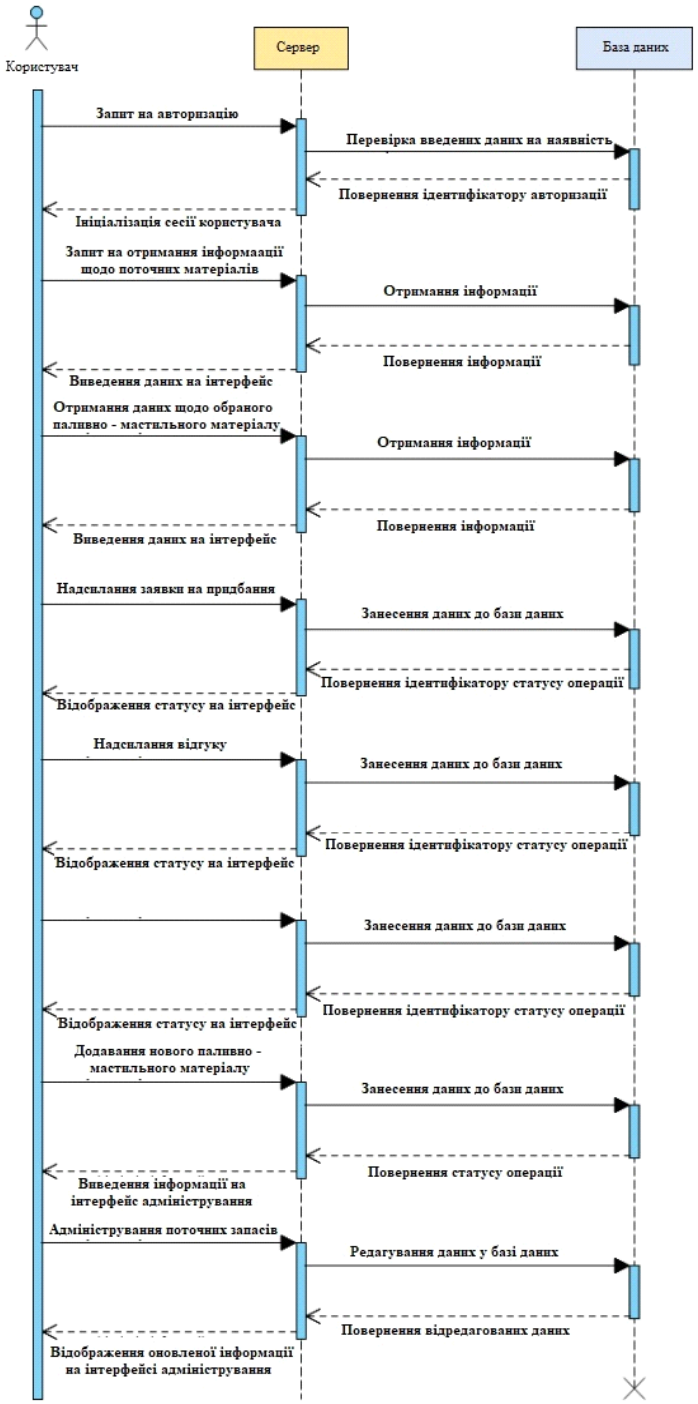


Рисунок 3.1 – Діаграма основних процесів взаємодії користувача з базою даних

Застосунок складається з набору програмних модулів, кожний з яких виконує одну чи декілька функцій. У таблиці 3.1 наведено призначення кожного проектованого модулю.

Таблиця 3.1 - Призначення програмних модулів

№	Позначення	Призначення
1	Index	Модуль головної сторінки інформаційної системи, відповідає за навігацію по застосунку, відображення інформації актуальних послуг застосунку.
2	Booking	Модуль для реалізації логіки взаємодії з базою даних для отримання інформації щодо обраної послуги.
3	Contact-us	Модуль для реалізації логіки залишення користувачем відгуку щодо використання.
5	Auth	Модуль для реалізації сторінки та логіки авторизації користувача у застусунку. Авторизація обов’язкова для використання.
6	Registration	Модуль для реалізації сторінки та логіки реєстрації користувача у застосунку. Реєстрація обов’язкова для використання.
7	Logout	Модуль для реалізації логіки завершеня сесії користувача.
9	Admin	Модуль для реалізації використання застосунку адміністратором з підвищеними правами доступу до бази даних.
10	Adminfeedbacks	Модуль для реалізації панелі керування адміністратора.
11	Adminfranchises	Модуль для реалізації панелі керування адміністратора.
12	Adminservices	Модуль для реалізації панелі керування адміністратора. Надає можливість адмініструвати базу даних застосунку.

3.3 Проектування бази даних

Коли основні процесів взаємодії користувача з базою даних визначено, перейдемо до проектування структури бази даних. У наведеній системі база даних використовується для того, щоб реалізувати весь функціонал для взаємодії з користувачем, а саме: збереження галереї зображень, збереження відгуків, збереження інформації щодо паливно-мастильних матеріалів тощо. Інтеграція бази даних потрібна для оптимізації роботи застосунку з критично важливими даними, що зберігаються під час використання користувачем.

Для початку реалізація бази даних необхідно обрати, яку саме СКБД буде використано для керування нею. Система керування базами даних MySQL може бути використана для реалізації засобів захисту даних та інформаційної безпеки з різних причин. Наведемо обґрунтування, чому саме MySQL може бути корисним для виконання поставленої мети роботи [18]:

- Керування доступом: MySQL надає розширені засоби керування доступом, такі як ролі, права доступу і автентифікація.
- Шифрування даних: MySQL підтримує різні методи шифрування даних, включаючи TLS/SSL для шифрування з'єднань між клієнтами і сервером, а також можливість шифрувати дані на рівні колонок або таблиць за допомогою різних криптографічних алгоритмів.
- Аудит і журналювання: MySQL надає можливості для створення аудиту та журналювання подій бази даних. Це дозволяє вам відстежувати, хто, коли і яким чином звертався до бази даних.
- Захист від SQL-ін'єкцій: MySQL дозволяє використовувати підготовлені запити та параметризовані запити, що допомагає запобігти атакам SQL-ін'єкції, що є принциповим пунктом відповідно до теми роботи.
- Запобігання втраті даних: MySQL може бути налаштований для автоматичного резервного копіювання даних.
- Засоби моніторингу: MySQL надає інструменти моніторингу, які дозволяють вам слідкувати за працездатністю сервера та бази даних, виявляти

незвичну активність та завчасно реагувати на потенційні загрози, що також є принциповим пунктом у реалізації захисту від DOS атак відповідно до теми роботи.

Система є веб-сервісом , тому дані будуть знаходитись у базі даних. Вони підтягуватимуться для подальшого відображення користувачеві, тому необхідно мати постійний зв'язок сервера з базою даних та клієнта із сервером.

Першим кроком реалізації бази даних є створення таблиці ідентифікаторів, яка наведена у таблиці 3.2. Лістинг створеної бази даних знаходиться у додатках (Додаток А).

Таблиця 3.2 – Ідентифікатори проектованої бази даних

Об'єкт	Власність	Тип	Розмірність	Ідентифікатор
Таблиця заявок на придбання паливо - мастильних матеріалів	Ідентифікатор	Varchar	100	bookingID
	Назва	Varchar	100	ServiceName
	Ім’я клієнта	Varchar	100	Name
	Прізвище клієнта	Varchar	100	Lname
	Телефон клієнта	Varchar	100	Phone
	Номер станції	Varchar	100	Station
	Номер колонки	Varchar	100	Column
	Кількість	Varchar	100	Amount
Таблиця відгуків	Ун. ідентифікатор	varchar	100	msgID
	Ім’я відправника	varchar	100	senderfname
	Прізвище відправника	varchar	100	senderlname
	Email відправника	varchar	100	sendereMail
	Відгук	Varchar	500	Senderfeedback
Таблиця заявок на франшизу	Ім’я	Varchar	100	Name
	Прізвище	Varchar	100	Specialisation
	Компанія	Varchar	100	Exp
	Позиція	Varchar	100	Practice

Продовження таблиці 3.2 – Ідентифікатори проектованої бази даних

Таблиця заявок на франшизу	Місто	Varchar	100	City
	Телефон	Varchar	100	Phone
	Email	Varchar	100	Email
Таблиця паливо - мастильних матеріалів	Ун. ідентифікатор	Varchar	100	serviceId
	Зображення	Varchar	100	serviceimg
	Назва	Varchar	100	serviceTitle
	Ціна за літр	Varchar	100	serviceprice
	Наявна кількість	Varchar	100	serviceamount
	Ступінь зтиснення	Varchar	100	Servicecompress
	Виробник	Varchar	100	Servicehost
	Короткий опис	Varchar	100	Servicedescr
Таблиця користувачів	Ун. ідентифікатор	Varchar	100	ID
	Логін	Varchar	100	Login
	Ім’я	Varchar	100	Name
	Прізвище	Varchar	100	Lname
	Пароль	Varchar	100	Password
	Країна	Varchar	100	Country
	Рівень доступу	Varchar	100	Lvl
Таблиця логів бази даних	Ун. ідентифікатор	Varchar	100	ID
	IP-адреса	Varchar	100	IP
	Дата та час запиту	Varchar	100	Created_at

Для візуалізації загальної структури бази даних, реалізуємо схему таблиць (Рисунок 3.2).

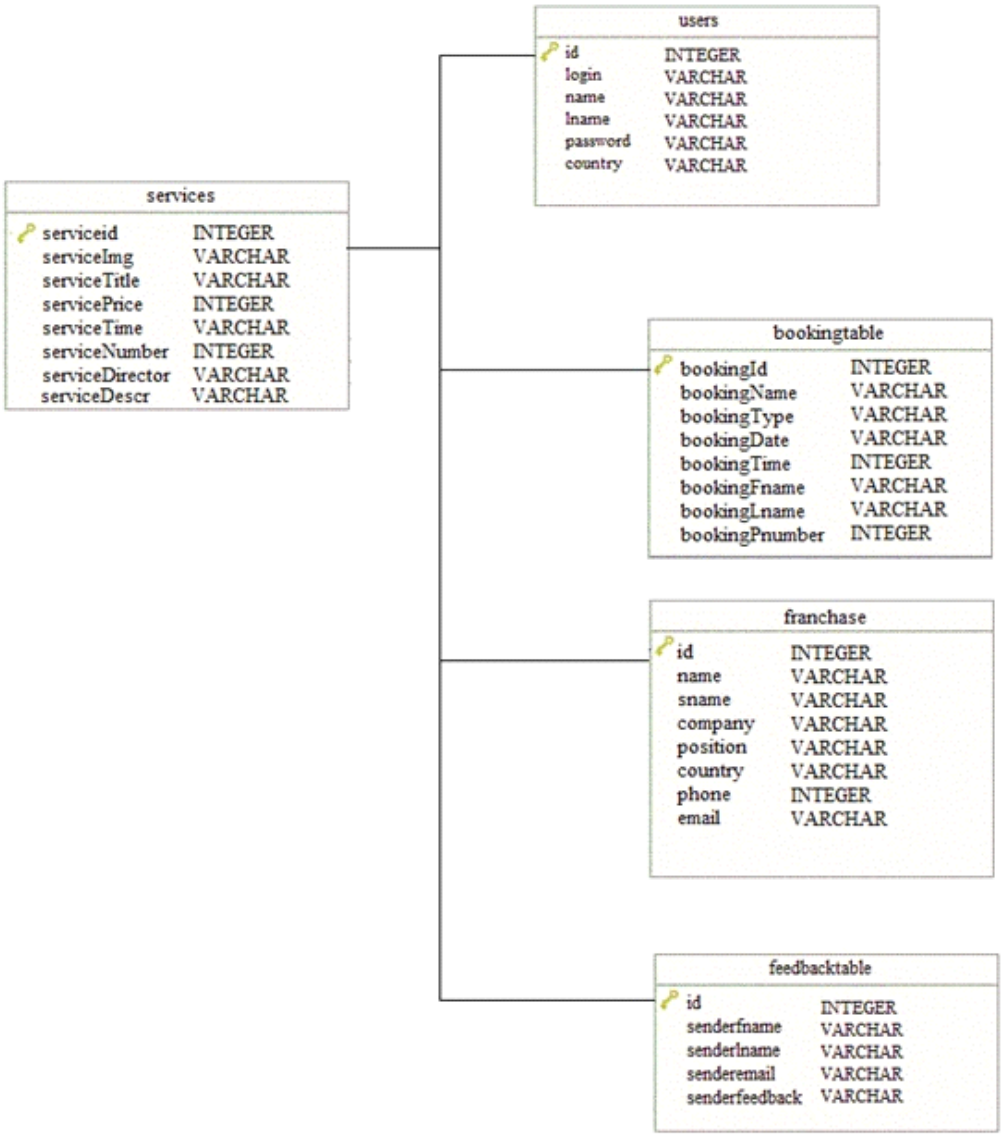


Рисунок 3.2 – Схема таблиць реалізованої бази даних

3.4 Реалізація базових засобів безпеки бази даних

Перш ніж переходити до безпосередньо реалізації захисту бази даних від SQL – ін’єкцій та DOS атак, необхідно реалізувати основні засоби безпеки цієї бази даних, а саме шифрування даних та розмежування рівню доступу до цих даних, оскільки якщо доступ до даних може отримати кожний користувач системи, або

вони зберігаються в незашифрованому вигляді – реалізація більш складних засобів безпеки є недоречною.

У будь якій базі даних одним з основних критеріїв безпеки даних на рівні з цілісністю є конфіденційність. Доступ до особистих даних користувачів не може мати не тільки зловмисник, а й персонал, що обслуговує базу даних. Для забезпечення інформаційної безпеки особистих даних користувачів, необхідно реалізувати шифрування цих даних за допомогою алгоритмів симетричного шифрування.

Симетричне шифрування є засобом безпеки даних, що передбачає використання одного й того самого ключа як для процесу шифрування, так і для процесу розшифрування. До симетричних алгоритмів застосовуються дві основні вимоги: повна втрата всіх статистичних закономірностей в об'єкті шифрування та відсутність лінійності.

Для проектованого застосунку буде реалізовано алгоритм шифрування, що базується на стандарті AES-256-CBC (Advanced Encryption Standard 256-bit Cipher Block Chaining) і використовує два секретні ключі. Цей алгоритм служить для забезпечення конфіденційності даних і використовує симетричний блочний шифр AES.

У реалізації алгоритму дані розбиваються на фіксовані блоки розміром 128 біт (16 байт). Кожен блок даних шифрується незалежно від інших блоків, і перед шифруванням до поточного блоку додається попередній блок даних. Це створює залежність між блоками, що підвищує стійкість алгоритму до атак.

Алгоритм використовує 256-бітний ключ, що робить його дуже стійким до спроб перебору. Кожен зашифрований блок даних служить як вхідний вектор для наступного блоку. Перший блок даних (вектор ініціалізації) може бути випадковим або згенерованим унікальним чином для кожного повідомлення або сесії шифрування.

Після шифрування всі блоки даних зв'язуються між собою і можуть бути збережені або передані разом з шифрованими даними. Розшифрування відбувається

в зворотному порядку, де кожен блок даних розшифровується і попередній блок додається до результату для відновлення оригінальних даних.

```
function encrypt_decrypt($string, $action = 'encrypt')
{
    $encrypt_method = "AES-256-CBC";
    $secret_key = 'diplomaproject';
    $secret_iv = ' diplomaproject ';
    $key = hash('sha256', $secret_key);
    $iv = substr(hash('sha256', $secret_iv), 0, 16); // алгоритм шифрування
    if ($action == 'encrypt') {
        $output = openssl_encrypt($string, $encrypt_method, $key, 0, $iv);
        $output = base64_encode($output);
    } else if ($action == 'decrypt') {
        $output = openssl_decrypt(base64_decode($string), $encrypt_method,
$key, 0, $iv);
    }
    return $output;
```

Рисунок 3.3 – Програмну реалізація розробленого шифрування

Реалізована функція має можливість працювати в режимі шифрування або розшифрування в залежності від параметру, отриманого на вхід. Такий метод реалізації є зручним у використанні, так як надає можливість надавати на вхід однієї функції як дані користувача, що потребують шифрування перед занесенням у базу даних, так і вже зашифровані дані з бази даних, що потребують дешифрування для взаємодії всередені системи.

Даний метод надає максимальну безпечність даним, так як навіть у випадку коли зловмисник якимось методом отримає доступ до бази даних, все, що він зможе там побачити – набір символів, який неможливо буде розшифрувати без володіння ключем шифрування. Даний метод в той же час вирішує також проблему безпеки з персоналом, що адмініструє базу даних, який таким самим чином не зможе отримати вигоду, реалізувавши особисті дані користувачів.

У системи, базу даних якої було реалізовано є необхідність взаємодії з даними різних груп користувачів. Звичайний користувач може взаємодіяти з базою даних лише на рівні отримання інформації щодо товарних залишків компанії та можливості додавати інформацію щодо свого замовлення до бази даних. У той же

час співробітники АЗС мають мати більші можливості взаємодії з базою даних, тому, що існує потреба в адмініструванні поточних товарних запасів, додавання нових, видалення існуючих, взаємодії зі зверненнями користувачів, інформація щодо яких також знаходиться у базі даних. Надання доступу до цього функціоналу звичайному користувачу є категорично неможливою, тому принципово важливо реалізувати другий рівень доступу з іншими правами на адміністрування бази даних. Так як даний рівень доступу надає можливість вносити критичні зміни, один фактор автентифікації при авторизації такого користувача є дуже небезпечним та не прийнятним в умовах нашого завдання. Щоб забезпечити необхідний рівень безпеки – реалізуємо додатковий фактор автентифікації користувача з підвищеними правами доступу до адміністрування за допомогою електронного листа.

Першим фактором автентифікації як і в звичайного користувача реалізуємо перевірку на наявність збігу введених користувачем автентифікаційних даних з наявними даними у базі даних.

```
    }else if(isset($_POST['adminlogin'])){
    $login = $_POST['adminlogin'];
    $password = $_POST['password'];
    $sql = "SELECT id FROM users WHERE role = 'admin' and login = '$login'";
    $check = "SELECT id FROM users WHERE role = 'admin' and login = '$login'
and password = '$password'";
    $result = mysqli_query($link,$sql);
    $res = mysqli_query($link,$check);
    if(mysqli_num_rows($result)==0) {
        $alert = "Такого користувача не існує, будь ласка, зареєструйтесь!";
    }elseif(mysqli_num_rows($res)==0){
        $alert = "Введений пароль невірний, спробуйте знову";
    }else{
        $fullurl = 'check.php?login=' . $login;
        echo "<script> window.location.href =  '$fullurl' </script>";
    };
```

Рисунок 3.4 – Реалізація перевірки на наявність збігу

У даній частині коду система отримує відповідні дані авторизації адміністратора, зберігає їх до змінних та виконує підключення до бази даних. Наступним кроком виконується перевірка, чи існує користувач з наведеним логіном у таблиці адміністраторів. Якщо такого користувача не зареєстровано, застосунок виводить на інтерфейс користувача ідентифікатор першої помилки. Наступним кроком виконується перевірка чи співпадає введений логін з введеним паролем, у випадку якщо вони не співпадають, застосунок виводить на інтерфейс користувача

ідентифікатор другої помилки. Якщо логін та пароль співпадають, система перенаправить користувача до другого фактору авторизації, а саме підтвердження автентифікації адміністратора за допомогою email токену.

Програмна реалізація даного фактору наведена далі:

```
require_once 'PHPMailer/src/Exception.php';
require_once 'PHPMailer/src/PHPMailer.php';
require_once 'PHPMailer/src/SMTP.php';
$url = $_GET['url'];
$login = $_GET['login'];
$link = mysqli_connect("localhost", "root", "", "azs");
$sql = "SELECT email FROM users WHERE login = '$login'";
$result = mysqli_query($link,$sql);
$myrow = mysqli_fetch_array ($result);
$to = $myrow[0];
$number = mt_rand(11111111,99999999);
$mail = new PHPMailer;
$mail->CharSet = 'UTF-8';
$mail->isSMTP();
$mail->SMTPAuth = true;
$mail->SMTPDebug = 0;
$mail->Host = 'ssl://smtp.gmail.com';
$mail->Port = 465;
$mail->Username = 'jonnytroy24@gmail.com';
$mail->Password = 'gtkrdaiqvsediucp';
$mail->setFrom('jonnytroy24@gmail.com', '');
$mail->addAddress($to)
$mail->Subject = 'Подтверждение входа';
$body = "Ваш код - " . $number;
$mail->msgHTML($body);
$mail->addAttachment(__DIR__ . '/image.jpg');
$mail->send();
```

Рисунок 3.5 – Реалізація перевірки на наявність збігу

У даній частині коду на електронну пошту адміністратора буде відправлено тимчасовий токен для підтвердження авторизації. Адреса електронної пошти витягується з бази даних адміністраторів, тому умовний зловмисник під час спроби авторизуватися під акаунтом адміністратора не буде навіть мати можливості отримати інформацію на яку електронну пошту було надіслано токен.

Якщо введений токен є ідентичним відправленому, користувача буде авторизовано у системі з додатковими правами адміністрування.

3.6 Реалізація захисту від SQL-ін'єкцій

Коли базу даних та інтерфейси взаємодії реалізовано, доречно буде перейти до безпосередньо реалізації засобів їх захисту. Створений інтерфейс має значну кількість полів типу input для взаємодії з користувачем, тому першим кроком реалізуємо їх безпеку від SQL-ін'єкцій, щоб не дати можливість умовному

зловмиснику впроваджувати шкідливий код для виконання операцій з даними на стороні бази даних.

Для реалізації цієї задачі найбільш доречно буде використовувати принцип екранування даних у SQL запиті на етапі обробки запиту користувача перед взаємодією з базою даних, а саме процес обробки вхідних даних перед включенням їх до SQL-запитів з метою запобігання атакам SQL ін'єкції. Головна ідея полягає в тому, щоб переконатися, що дані введені користувачем не інтерпретуються як частина SQL-запиту, а розглядаються як звичайний текст. Для того, щоб забезпечити надійний рівень безпеки, використаємо 2 принципи екранування, а саме параметризовані запити та використання вбудованих функцій екранування мови програмування PHP. Реалізуємо принцип екранування на прикладі процесу авторизації користувача.

Першим кроком виконаємо з’єднання з базою даних, отримаємо безпосередньо самі дані користувача, які було відправлено методом POST для перевірки їх наявності у базі даних для виконання авторизації.

Метод POST є одним із стандартних методів HTTP (Hypertext Transfer Protocol) і використовується для надсилання даних на веб-сервер з метою створення нового ресурсу або оновлення існуючого ресурсу на сервері. Метод POST використовується для взаємодії з базою даних шляхом відправлення даних на сервер з метою створення, оновлення або видалення записів в базі даних. Основні етапи використання методу POST у нашому випадку взаємодії з базою даних включають:

- Формування запиту: Клієнт формує запит HTTP методом POST. Цей запит містить необхідні дані, які клієнт бажає відправити на сервер для обробки базою даних. Зазвичай ці дані включаються у вигляді параметрів або форми в тілі запиту.
- Обробка на сервері: Сервер приймає запит і обробляє дані, що надходять у тілі POST-запиту. Сервер може перевіряти, чи валідні дані, а також виконувати всі необхідні перевірки та операції перед взаємодією з базою даних.
- Взаємодія з базою даних: Після обробки даних сервер може використовувати SQL-запити для взаємодії з базою даних. Це може включати

вставку нових записів у таблиці бази даних, оновлення існуючих записів або видалення даних, в залежності від конкретного завдання.

- Обробка відповіді: Після взаємодії з базою даних сервер створює відповідь, яка може містити інформацію про результат виконаних дій. Ця відповідь може бути надіслана клієнту для подальшого відображення результатів або іншої обробки.

Коли дані користувача отримано, реалізуємо їх екранування за допомогою функції mysqli_real_escape_string.

Mysqli_real_escape_string - це функція, яка використовується для екранування рядків перед їхнім використанням в SQL-запитах в PHP, при взаємодії з MySQL в якості бази даних. Її головна мета полягає в запобіганні атакам SQL-ін'єкції шляхом заміни спеціальних символів в рядках на їхні екрановані еквіваленти, що дозволяє вставляти дані безпечно в SQL-запити.

Ключові аспекти функції mysqli_real_escape_string реалізують:

- Екранування спеціальних символів
- Використання підготовлених запитів
- Параметрізовані запити:
- Користь в уникненні SQL-ін'єкцій

Однак важливо відзначити, що використання mysqli_real_escape_string не є повністю безпечним для всіх сценаріїв і може показувати максимальну ефективність тільки у комбінації з використанням підготовлених запитів. Найкращим підходом до безпеки баз даних є використання підготовлених запитів та обрання правильного методу екранування в залежності від контексту і типу даних.

```
$link = mysqli_connect("localhost", "root", "", "azs");
if (isset($_POST['submit'])) {
    // Екранування вхідних даних
    $tyy = mysqli_real_escape_string($link, $_POST['name']);
    $pass = mysqli_real_escape_string($link, $_POST['position']);
```

Рисунок 3.6 – Реалізація екранування

Дані екрановано, але такий засіб все ще не задовольняє нас своєю безпечністю, тому підготовимо параметризовані запити, в яких значення параметрів будуть відокремлені від самого SQL-коду. Спочатку реалізуємо запит для вибору користувача за логіном.

```
$sql = "SELECT id FROM users WHERE login = ?";
$stmt = mysqli_prepare($link, $sql);
mysqli_stmt_bind_param($stmt, "s", $tyy);
mysqli_stmt_execute($stmt);
$result = mysqli_stmt_get_result($stmt);
```

Рисунок 3.7 – Реалізація запиту для вибору користувача за логіном

Таким саме чином за аналогією реалізуємо запит для перевірки логіну та паролю.

```
$check = "SELECT id FROM users WHERE login = ? and password = ?";
$stmt = mysqli_prepare($link, $check);
mysqli_stmt_bind_param($stmt, "ss", $tyy, $pass);
mysqli_stmt_execute($stmt);
$res = mysqli_stmt_get_result($stmt)
```

Рисунок 3.8 – Реалізація запиту для перевірки логіну та паролю

Коли запити сформульовано, реалізуємо безпосередньо обробку відповіді, що буде виконувати авторизацію користувача та ініціювати його сесію якщо запит виконано успішно, або виводити на інтерфейс користувача помилку якщо запит повернув негативний результат.

```
if (mysqli_num_rows($result) == 0) {
        echo "Такого користувача не існує, будь ласка, зареєструйтесь!";
    } elseif (mysqli_num_rows($res) == 0) {
        echo "Введений пароль невірний, спробуйте знову";
    } else {
        // Успішний вхід, встановлення сесії та перенаправлення
        $_SESSION['itlogin'] = $tyy;
        header("location: /index.php");
    }
```

Рисунок 3.9 – Реалізація обробки відповіді

Реалізований метод захисту дозволяє бути впевненим, що будь які дані, введені користувачем, не будуть виконані на стороні бази даних в якості команди.

3.7 Реалізація захисту від атак типу DoS

Наступним важливим критерієм є захист від атак типу DoS. Для вирішення цієї проблеми доречно буде реалізувати функціонал, що буде надавати обмеження на запити до нашого застосунку та бази даних. Перш ніж приступати до написання коду, необхідно визначитись з принципом роботи алгоритму обмеження. Найбільш зручним варіантом буде взаємодія з базою даних для збереження там даних щодо останніх запитів до застосунку, перевірка кількості запитів від актуальної IP – адреси користувача та обмеження доступу для наведеної адреси, якщо кількість запитів перевищую встановлений ліміт.

Почнемо реалізацію з з'єднання з базою даних та отримання кількості запитів за визначений інтервал від актуального користувача за його IP – адресою. Для забезпечення безпеки взаємодії з базою даних будемо використовувати параметризовані запити за аналогією до розроблених у попередньому підрозділі.

```
$connection = mysqli_connect("localhost", "root", "", "azs");
$query = "SELECT COUNT(*) as request_count FROM request_log WHERE ip = ? AND
created_at >= NOW() - INTERVAL ? SECOND";
$stmt = mysqli_prepare($connection, $query);
mysqli_stmt_bind_param($stmt, "si", $ip, $interval);
mysqli_stmt_execute($stmt);
$result = mysqli_stmt_get_result($stmt);
$row = mysqli_fetch_assoc($result);
```

Рисунок 3.10 – Реалізація з'єднання з базою даних

Коли інформація щодо кількості запитів отримана, реалізуємо безпосередньо перевірку перевищено ліміт запитів від IP-адреси поточного користувача чи ні.

```
if ($row["request_count"] >= $limit) {
    return false; // Перевищено ліміт запитів
}
```

Рисунок 3.11 – Реалізація перевірки

Якщо ліміт не перевищено, необхідно зробити запис в лог бази даних, щоб поточний запит був врахований під час наступного підрахунку кількості запитів. Також необхідно повернути параметр «True», щоб мати можливість обробити результат запиту.

```
$insertQuery = "INSERT INTO request_log (ip) VALUES (?)";
$stmt = mysqli_prepare($connection, $insertQuery);
mysqli_stmt_bind_param($stmt, "s", $ip);
mysqli_stmt_execute($stmt);
return true; // Запит допущений
```

Рисунок 3.12 – Запис в лог бази даних

Логіку перевірки реалізовано, для її коректної роботи визначимо IP-адресу поточного користувача, ліміт на кількість запитів та інтервал, під час якого можна виконати визначену кількість запитів.

```
$ip = $_SERVER['REMOTE_ADDR'];
$limit = 10; // Максимальна кількість запитів
$interval = 60; // Інтервал в секундах
```

Рисунок 3.13 – Визначення певних параметрів

Останнім кроком необхідно визвати реалізовану функцію та обробити результат її виконання. Якщо кількість поточних запитів знаходиться у необхідному інтервалі – користувача необхідно перенести на сторінку застосунку, в іншому випадку – вивести на інтерфейс користувача інформацію про помилку.

```
if (checkRequestLimit($ip, $limit, $interval)) {
    // Виконати запит
        header("location: /index.php");
    } else {
    // Повідомлення про перевищення ліміту запитів
    echo "<script>alert('Ви перевищили ліміт запитів. Спробуйте
пізніше.')</script>";
}
```

Рисунок 3.14 – Виклик реалізованої функції та обробка результату її виконання

Реалізований метод захисту є гарантією, що умовний зловмисник не зможе перевищити кількість запитів, яку база даних та застосунок можуть обробити.

3.8 Тестування реалізованих засобів безпеки

Коли всі необхідні засоби захисту реалізовано, необхідно провести їх тестування, щоб впевнитись у тому, що реалізований функціонал повністю виконує поставлені перед ним завдання. Протестуємо реалізований функціонал за допомогою виконання наступних кроків:

- Тестування процесу шифрування даних користувача при занесенні до Бази даних.
- Тестування процесу дешифрування даних з бази даних для взаємодії з системою.
- Спроба отримати доступ до взаємодії з базою даних неавторизованому користувачу;
- Спроба впровадити SQL – запит до бази даних через поля форм;
- Спроба DoS атаки на застосунок та базу даних;
- Спроба автентифікації користувача з підвищеними правами доступу до адміністрування бази даних;

Першим протестуємо процес шифрування особистих даних користувача, бо він є принциповим і пов’язаним зі всіма іншими. При некоректній роботі процесу шифрування та дешифрування система не зможе взаємодіяти з даними, які зберігаються у базі даних у зашифрованому вигляді.

Проведемо тестування на прикладі реєстрації користувача (Додаток Г). Перейдемо на сторінку реєстрації та заповним форму реєстрації.

В результаті система видала повідомлення про успішну реєстрацію, тобто дані було зашифровано та занесено до бази даних.

Щоб впевнитись у вдалому шифруванні перейдемо до інтерфейсу взаємодії з базою даних PhpMyadmin та подивимось на результат, який було занесено у колонку паролю користувача.

Як можна побачити, дані користувача, які було подано до функції шифрування перед збереженням у базі даних, а саме пароль користувача успішно зашифровано впровадженим алгоритмом шифрування.

Наступним кроком виконаємо перевірку на коректність роботи зворотнього процесу, тобто дешифрування даних (Додаток Д). Для перевірки спробуємо авторизуватися під звичайними логіном та паролем, під якими було проведено авторизацію.

Як результат отримуємо успішну авторизацію і перенаправлення користувача до головної сторінки застосунку, тобто система успішно виконала процес дешифрування паролю з бази даних, порівняння його з паролем в звичайному вигляді та авторизацію користувача.

Наступним протестуємо надійність процесу автентифікації користувача (Додаток Д). Припустимо, що умовний зловмисник заволодів даними з бази даних та намагається отримати доступ до облікового запису користувача. Спробуємо авторизуватися у застосунку, використавши безпосередньо ті дані, що зберігаються у зашифрованому вигляді.

Як можна побачити, в результаті, система виводить на інтерфейс користувача помилку, та не створює ідентифікатор сесії, тим самим не надаючи доступ до облікового запису користувача.

Наступним кроком перевіримо стійкість впроваджених засобів безпеки проти SQL – ін’єкцій, спробувавши впровадити ін’єкцію за допомогою поля взаємодії з користувачем (Додаток Е). Впровадимо ін’єкцію з метою повного видалення таблиці користувачів системи з усіма даними та протестуємо її результати з реалізованими засобами безпеки та без них.

Результати тестування наочно демонструють наскільки важливими є засоби захисту від SQL-ін’єкцій, оскільки навіть така проста ін’єкція може повністю видалити всі дані з бази даних, або нашкодити їм. В той самий час реалізовані засоби захисту заблокували дію ін’єкції та повністю виконали поставлене завдання.

Наступним кроком протестуємо надійність впровадженого алгоритму протидії DoS атакам (Додаток Ж). Для цього виконаємо цільову дію, а саме звернемось до застосунку, передавши йому дані для взаємодії з базою даних 20 разів одночасно.

Як результат – на перших 10 вкладинках браузера маємо успішно завантажену сторінку, на 10 – 20 вкладинці система визначила перевищення ліміту запитів з

нашої IP – адреси, вивела на інтерфейс користувача повідомлення про помилку та призупинила виконання скрипту, тобто процес взаємодії з базою даних не був ініціалізований, що надає можливість користувачу виконати лише ту кількість запитів, яку база даних може обробити без збоїв у роботі та гарантує захист від атак типу DoS.

Проведемо те саме тестування, відправивши одночасно 1000 запитів до застосунку для отримання конкретних числових значень його швидкодії з впровадженим алгоритмом безпеки від DOS атак та без нього. Результат (Рисунок 3.3):

З впровадженим алгоритмом безпеки:

Середній час завантаження сторінки: 1.4 секунди.

Обсяг запитів на сервер: 1000 запитів.

Відсоток успішно обслужених запитів: 100% (без відмов).

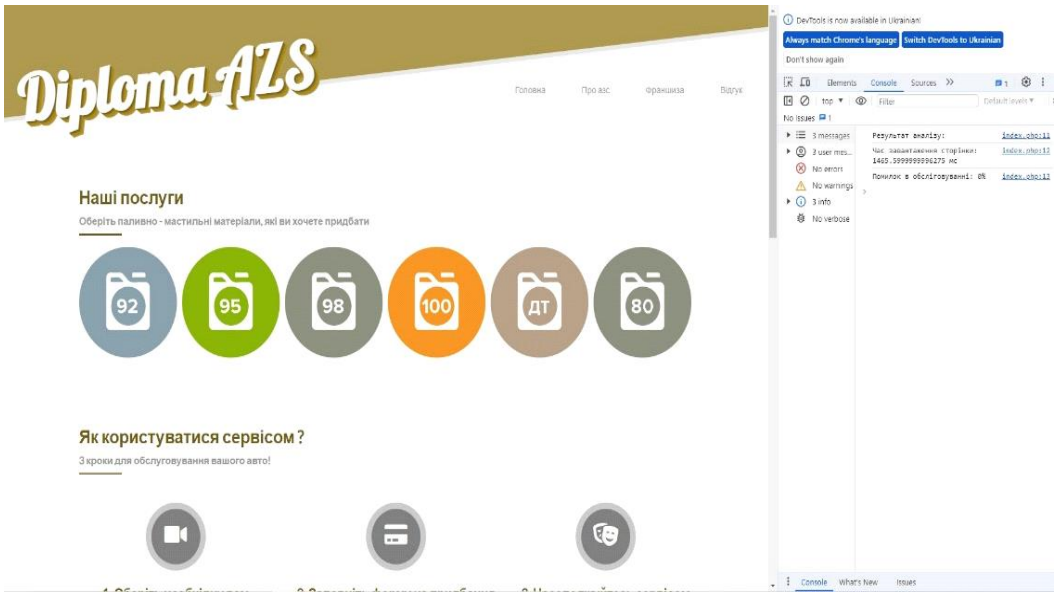


Рисунок 3.15 – Результат тестування застосунку з провадженими засобами безпеки

Без засобів безпеки (Рисунок 3.4):

Середній час завантаження сторінки: 20 секунд.

Обсяг запитів на сервер: 1000 запитів.

Відсоток успішно обслужених запитів: 53.2% (з відмовами в обслуговуванні для половини запитів).

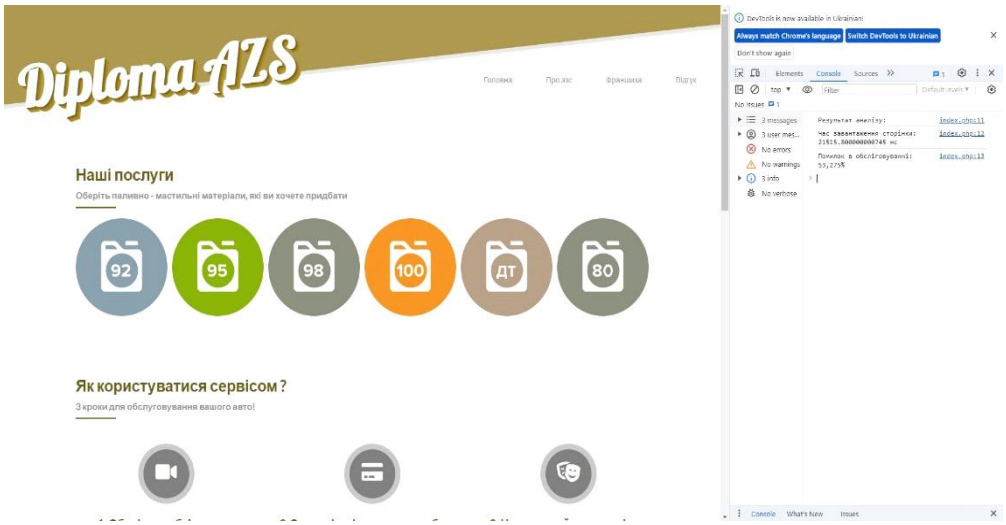


Рисунок 3.16 – Результат тестування застосунку без впроваджених засобів безпеки

Останнім кроком протестуємо процес автентифікації користувача з підвищеними правами доступу до адміністрування бази даних (Додаток І). Перший фактор автентифікації є аналогічним до звичайного користувача, тому одразу перейдемо до тестування другого, а саме підтвердження автентифікації за допомогою email токену. Для цього введемо у форму авторизації логін та пароль користувача, який має права адміністрування бази.

Якщо цей фактор авторизації пройдено успішно, система створить тимчасовий токен, що буде відправлено на електонну пошту, яку було вказано під час створення адміністративних прав цього користувача.

Протестуємо спробу входу з вірним токеном, що було згенеровано системою та з свідомо невірним щоб визначити коректність функціоналу.

Як можна побачити на першому результаті система успішно автентифікувала користувача с підвищеним рівнем редагування даних у базі даних та надала йому доступ до інтерфейсу адміністрування даних, на прикладі предметної галузі – залишків товарних запасів, звернень та відгуків користувачів.

На другому прикладі можна побачити, що система співставила невірний токен зі згенерованим, визначила спробу несанкціонованого доступу та відмовила користувачу в отримані додаткових прав доступу до даних у базі даних.

З отриманих результатів можна зробити висновок, що всі реалізовані засоби захисту застосунку та бази даних працюють коректно та виконують поставлені завдання.

Узагальнивши результати роботи можна визначити, що впровадження реалізованих засобів безпеки є критично важливим для застосунку та при комплексному використанні робить його достатньо надійним як зі сторони захисту даних від різноманітних ін’єкцій, так і від перевантажень та відмов в обслуговуванні через атаки типу DOS. Реалізація застосунку та бази даних без відповідних засобів безпеки категорично не рекомендується, бо може призвести як до втрати або пошкодження конфіденційних даних, так і до відмови в обслуговуванні потенційного користувача.

3.9 Висновок до розділу

У даному розділі було виконано проектування та практична реалізація засобів захисту бази даних автоматизованої системи автозаправної станції. Проведено визначення вимог до безпеки бази даних, що проектується та проектування структури застосунку для взаємодії з базою даних. Спроектовано базу даних, що задовольняє роботі процесів обраної предметної галузі. Проведено реалізацію проектованого застосунку для взаємодії з базою даних за допомогою мов програмування PHP, JavaScript, HTML, CSS. Реалізовано захист від SQL-ін'єкцій за допомогою методів параметризації та екранування. Проведено реалізацію захисту застосунку та бази даних від атак типу DoS та реалізацію процесів шифрування та дешифрування даних у базі даних. Реалізовано розмежування прав доступу та додаткового фактору автентифікації користувачів. Результати тестування показують, що реалізовані засоби безпеки повністю виконують свої функції та можуть бути використані для протидії вразливостям баз даних.

ВИСНОВОК

В процесі виконання роботи були отримані наступні результати:

Проведено аналіз вразливостей баз даних та засоби їх захисту. Визначено основні вразливості бази даних. Проаналізовано та визначено вразливості на різних рівнях, а саме рівні проектування, реалізації та конфігурації. Проведено аналіз основних принципів безпеки баз даних. Сформульовано основні принципи реалізації комплексної безпеки бази даних на рівнях застосунку, бази даних та мережі.

Спроектовано та реалізовано базу даних та застосунок для взаємодії з нею, що задовольняє потребам предметної галузі. На основі проведеного аналізу та визначених вразливостей спроектовано та програмно реалізовано засоби захисту даних у базі даних від визначених загроз. Створені методи забезпечують наступні рівні захисту даних:

1) Надання доступу до будь якої взаємодії з даними лише користувачам, що пройшли процес автентифікації;

2) Реалізація розмежування прав доступу до різних частин бази даних для різних груп користувачів;

3) Реалізація процесів шифрування даних розробленим алгоритмом шифрування при занесенні їх до бази даних та дешифрування цих даних при відображенні їх користувачу;

4) Надання засобів захисту від SQL-ін'єкцій екрануванням та параметризованими запитами;

5) Забезпечення багатофакторної автентифікації для користувачів, які мають підвищений рівень доступу до даних;

6) Забезпечення захисту від атак типу DoS за рахунок відстеження сесій користувача та лімітування його запитів до системи;

Проведене тестування показало, що реалізовані засоби захисту повністю виконують визначену задачу та надають комплексний рівень захисту даних у базі

даних, що доводить успішність виконання завдання та можливість використання розробленої системи для вирішення поставленої задачі.

ПЕРЕЛІК ДЖЕРЕЛ ТА ПОСИЛАННЯ

1. Abraham S., Henry F. Korth, S. Sudarshan Database System Concepts 7th Edition. NY : McGraw-Hill Education, 2019 1376c.
2. Iqbal A., Khan S., Niazi M., Humayun M., Sama N., Khan A., Ahmad A. Advancing database security: a comprehensive systematic mapping study of potential challenges. Wireless Netw. 2023. P. 1–28.
3. Connolly T., Begg C. Database Systems: A Practical Approach to Design, Implementation, and Management. (Sixth Edition) Boston: Addison-Wesley, 2014. 1440 p.
4. Вілігура В. В. Систематизація загроз і вразливостей характерних для баз даних і СУБД. Праці 7-ої Міжнародній науково-технічної конференції «Комп'ютерне моделювання в наукоємних технологіях (КМНТ-2021), 21-23 квітня 2021 р. Харків: Харківський національний університет імені В. Н. Каразіна, 2021. С.83–86.
5. Грайворонський М.В. Сучасні підходи до забезпечення кібернетичної безпеки [Текст] / Матеріали XVII Всеукраїнської науковопрактичної конференції студентів, аспірантів та молодих вчених “Теоретичні і прикладні проблеми фізики, математики та інформатики”, НТУУ “КПІ”, 2015 р..
6. Justin Clarke SQL Injection Attacks and Defense. Second Edition. Elsevier, 2012. 547 p.
7. "CISSP Study Guide," 8th Edition. Мільєн, Майкл, та Чапман, Джеймс. Indianapolis: Wiley, 2018. 1104 с.
8. "Cyberpower and National Security: Policy Recommendations for a Strategic Framework," in Cyberpower and National Security, FD Kramer, S. Starr, [Текст] L.K. Wentz (ed.), National Defense University Press, Washington (DC) 2009.
9. ISO/IEC 27032:2012 [Текст] Information technology — Security techniques — Guidelines for cybersecurity

10. Vacca, J. R. "Computer and Information Security Handbook," 2nd Edition. New York: McGraw-Hill, 2013. 1104 p.
11. Conklin, W. A., White, G., Williams, D., Cothren, C., Davis, R. L. "Principles of Computer Security: CompTIA Security+ and Beyond," 5th Edition. New York: McGraw-Hill, 2014. 800 p.
12. Erickson, J. "Hacking: The Art of Exploitation." San Francisco: No Starch Press, 2008. 488 p.
13. Stallings, W. "Network Security Essentials: Applications and Standards," 6th Edition. Boston: Pearson, 2017. 608 p.
14. Stallings, W. "Cryptography and Network Security: Principles and Practice," 7th Edition. Boston: Pearson, 2016. 752 p.
15. Stuttard, D., Pinto, M. "The Web Application Hacker's Handbook: Finding and Exploiting Security Flaws." Indianapolis: Wiley, 2011. 912 p.
16. Schneier, B. "Applied Cryptography: Protocols, Algorithms, and Source Code in C," 2nd Edition. New York: Wiley, 1996. 784 p.
17. NIST Special Publication 800-53, "Security and Privacy Controls for Federal Information Systems and Organizations." National Institute of Standards and Technology (NIST), 2020.
18. Anderson, R. "Security Engineering: A Guide to Building Dependable Distributed Systems," 2nd Edition. Indianapolis: Wiley, 2020. 1060 p.

ДОДАТОК А
Лістинг бази даних

```
-- phpMyAdmin SQL Dump
-- version 5.2.0
-- https://www.phpmyadmin.net/
--
-- Хост: 127.0.0.1:3306
-- Час створення: Лис 27 2023 р., 15:41
-- Версія сервера: 8.0.30
-- Версія PHP: 8.1.9

SET SQL_MODE = "NO_AUTO_VALUE_ON_ZERO";
START TRANSACTION;
SET time_zone = "+00:00";


/*!40101 SET @OLD_CHARACTER_SET_CLIENT=@@CHARACTER_SET_CLIENT */;
/*!40101 SET @OLD_CHARACTER_SET_RESULTS=@@CHARACTER_SET_RESULTS */;
/*!40101 SET @OLD_COLLATION_CONNECTION=@@COLLATION_CONNECTION */;
/*!40101 SET NAMES utf8mb4 */;

--
-- База даних: `azs`
--

-- --------------------------------------------------------

--
-- Структура таблиці `bookingtable`
--

CREATE TABLE `bookingtable` (
  `bookingID` int NOT NULL,
  `ServiceName` varchar(100) CHARACTER SET utf8mb4 COLLATE utf8mb4_general_ci DEFAULT
NULL,
  `Name` varchar(100) CHARACTER SET utf8mb4 COLLATE utf8mb4_general_ci DEFAULT NULL,
  `Lname` varchar(100) CHARACTER SET utf8mb4 COLLATE utf8mb4_general_ci DEFAULT NULL,
  `Phone` varchar(50) CHARACTER SET utf8mb4 COLLATE utf8mb4_general_ci DEFAULT NULL,
  `Station` varchar(50) CHARACTER SET utf8mb4 COLLATE utf8mb4_general_ci DEFAULT
NULL,
  `Col` varchar(100) CHARACTER SET utf8mb4 COLLATE utf8mb4_general_ci DEFAULT NULL,
  `Amount` varchar(100) CHARACTER SET utf8mb4 COLLATE utf8mb4_general_ci DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

--
-- Дамп даних таблиці `bookingtable`
--

INSERT INTO `bookingtable` (`bookingID`, `ServiceName`, `Name`, `Lname`, `Phone`,
`Station`, `Col`, `Amount`) VALUES
(40, 'Дизельне пал.', 'Charles', 'Kayne', '4124142141', '3', 'Колонка 3', '22'),
(41, 'Бензин А-95', 'sgstgs', 'sdgdsg', 'dsgsggd', '4', 'Колонка 3', '23'),
(42, 'Бензин А-95', 'sdhsh', 'sdhhsd', 'sdhsh', '2', 'Колонка 4', '412414');

-- --------------------------------------------------------
```

```
--
-- Структура таблиці `feedbacktable`
--

CREATE TABLE `feedbacktable` (
  `id` int NOT NULL,
  `senderfName` varchar(50) NOT NULL,
  `senderlName` varchar(50) DEFAULT NULL,
  `sendereMail` varchar(100) NOT NULL,
  `senderfeedback` varchar(500) DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

--
-- Дамп даних таблиці `feedbacktable`
--

INSERT INTO `feedbacktable` (`id`, `senderfName`, `senderlName`, `sendereMail`,
`senderfeedback`) VALUES
(4, 'wf', 'etwt', 'email2@gmail.com', 'wetwt'),
(5, 'Alex', 'Dikkens', 'email1@gmail.com', 'sdggdg');

-- --------------------------------------------------------

--
-- Структура таблиці `franchase`
--

CREATE TABLE `franchase` (
  `id` int NOT NULL,
  `name` varchar(40) DEFAULT NULL,
  `sname` varchar(40) DEFAULT NULL,
  `company` varchar(40) DEFAULT NULL,
  `position` varchar(40) DEFAULT NULL,
  `city` varchar(40) CHARACTER SET utf8mb4 COLLATE utf8mb4_0900_ai_ci DEFAULT NULL,
  `phone` varchar(40) DEFAULT NULL,
  `email` varchar(40) DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;

-- --------------------------------------------------------

--
-- Структура таблиці `movietable`
--

CREATE TABLE `movietable` (
  `movieID` int NOT NULL,
  `movieImg` varchar(150) CHARACTER SET utf8mb4 COLLATE utf8mb4_general_ci NOT NULL,
  `movieTitle` varchar(100) CHARACTER SET utf8mb4 COLLATE utf8mb4_general_ci NOT
NULL,
  `movieGenre` varchar(500) CHARACTER SET utf8mb4 COLLATE utf8mb4_general_ci NOT
NULL,
  `movieDuration` varchar(500) CHARACTER SET utf8mb4 COLLATE utf8mb4_general_ci NOT
NULL,
  `movieRelDate` varchar(500) CHARACTER SET utf8mb4 COLLATE utf8mb4_general_ci NOT
NULL,
  `movieDirector` varchar(500) CHARACTER SET utf8mb4 COLLATE utf8mb4_general_ci NOT
NULL,
  `movieActors` varchar(1500) CHARACTER SET utf8mb4 COLLATE utf8mb4_general_ci NOT
NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

--
```

```
-- Дамп даних таблиці `movietable`
--

INSERT INTO `movietable` (`movieID`, `movieImg`, `movieTitle`, `movieGenre`,
`movieDuration`, `movieRelDate`, `movieDirector`, `movieActors`) VALUES
(1, 'img/img1.png', 'Бензин А-92', '45.59грн.', '814', '10-10.5 кг/см2', 'ВІКО-НАФТА,
ТОВ', 'Найпопулярніший через низьку вартість бензин у нашій країні. Насправді це
просте паливо, що не містить великої кількості присадок, призначене для карбюраторних
і деяких інжекторних моторів.'),
(2, 'img/img2.png', 'Бензин А-95', '46.66грн.', '475', '10,5 - 12 кг/см2', 'ЗАХІД
ГРУП ЮА, ТОВ', 'Коштує дорожче 92-го, але до преміального ще не дотягує. Вартість
виправдана появою невеликої кількості присадок та відсутністю свинцю у складі, що
робить вихлоп менш токсичним. АІ-95 стане ідеальною «їжею» для інжекторних
двигунів.'),
(3, 'img/img3.png', 'Бензин А-98', '79,99грн.', '450л.', '12-14 кг/см2', 'УКРАЇНСЬКА
НАФТОТРЕЙДІНГОВА КОМПАНІЯ, ТОВ', 'Цей вид палива відкриває лінійку \"преміальних\".
АІ-98 заливають власники форсованих двигунів. Нагадаємо, що форсованими називають ті
двигуни, які здатні видати максимальну потужність за рахунок високого тиску в камері
згоряння. Щоб двигун працював справно, доводиться розщедрюватися на АІ-98. Насправді
цей бензин використовується досить рідко. Основні споживачі – власники гоночних
автомобілів.'),
(4, 'img/img4.png', 'Бензин А-100', '112.44грн.', '176', '14+ кг/см2', 'ГРАНД
АВТОТРАНС, ТОВ', 'Найновіший вид палива на українському ринку. АІ з октановим числом
100 здатний суттєво знизити витрату палива. Кожні 100 кілометрів вдасться заощадити
10%. Звичайно, коштує він значно дорожче за інших і є універсальним. Щоправда,
найбільше це паливо підходить для турбованих двигунів.'),
(5, 'img/img5.png', 'Дизельне пал.', '47,55грн.', '600л.', '18-22 кг/см2',
'ЕНЕРГЕТИЧНІ РЕСУРСИ УКРАЇНИ, ТОВ', 'Рідкий продукт, що використовується як паливо у
дизельному двигуні внутрішнього згоряння. Зазвичай під цим терміном розуміють паливо,
що виходить із гасово-газойлевих фракцій прямої перегонки нафти.'),
(6, 'img/img6.png', 'Бензин А-80', '254грн.', '150л.', '8.5-9 кг/см2', 'Авер
Технолоджі, ТОВ', 'Продукт переробки нафти методом гідрокрекінгу, коксування з
додаванням мети-трет-бутилового ефіру та інших присадок. Сфера застосування бензину
цієї марки – забезпечення роботи двигунів внутрішнього згоряння легкового та
вантажного транспорту, сільськогосподарської техніки.');

-- --------------------------------------------------------

--
-- Структура таблиці `request_log`
--

CREATE TABLE `request_log` (
  `id` int NOT NULL,
  `ip` varchar(255) COLLATE utf8mb4_general_ci NOT NULL,
  `created_at` timestamp NULL DEFAULT CURRENT_TIMESTAMP
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci;

--
-- Дамп даних таблиці `request_log`
--

INSERT INTO `request_log` (`id`, `ip`, `created_at`) VALUES
(1, '127.0.0.1', '2023-10-12 03:38:24'),
(2, '127.0.0.1', '2023-10-12 03:38:25'),
(3, '127.0.0.1', '2023-10-12 03:38:26'),
(4, '127.0.0.1', '2023-10-12 03:38:26'),
(5, '127.0.0.1', '2023-10-12 03:38:26'),
(6, '127.0.0.1', '2023-10-12 03:38:27'),
(7, '127.0.0.1', '2023-10-12 03:38:27'),
(8, '127.0.0.1', '2023-10-12 03:38:27'),
(9, '127.0.0.1', '2023-10-12 03:38:27'),
(10, '127.0.0.1', '2023-10-12 03:38:27'),
```

```
(11, '127.0.0.1', '2023-10-12 03:39:43'),
(12, '127.0.0.1', '2023-10-12 03:39:44'),
(13, '127.0.0.1', '2023-10-12 03:39:45'),
(14, '127.0.0.1', '2023-10-12 03:39:46'),
(15, '127.0.0.1', '2023-10-12 03:39:47'),
(16, '127.0.0.1', '2023-10-12 03:39:47'),
(17, '127.0.0.1', '2023-10-12 03:39:48'),
(18, '127.0.0.1', '2023-10-12 03:39:48'),
(19, '127.0.0.1', '2023-10-12 03:39:49'),
(20, '127.0.0.1', '2023-10-12 03:39:50'),
(21, '127.0.0.1', '2023-10-12 05:21:50'),
(22, '127.0.0.1', '2023-10-12 05:25:41'),
(23, '127.0.0.1', '2023-10-12 05:25:46'),
(24, '127.0.0.1', '2023-10-12 05:26:35'),
(25, '127.0.0.1', '2023-10-12 05:26:39'),
(26, '127.0.0.1', '2023-10-12 05:27:06'),
(27, '127.0.0.1', '2023-10-12 05:57:24'),
(28, '127.0.0.1', '2023-10-12 05:57:38'),
(29, '127.0.0.1', '2023-10-12 05:57:48'),
(30, '127.0.0.1', '2023-10-12 05:58:36'),
(31, '127.0.0.1', '2023-10-12 05:58:39'),
(32, '127.0.0.1', '2023-10-12 05:58:46'),
(33, '127.0.0.1', '2023-10-12 06:10:37'),
(34, '127.0.0.1', '2023-10-12 06:13:42'),
(35, '127.0.0.1', '2023-10-12 06:15:01'),
(36, '127.0.0.1', '2023-10-12 06:15:03'),
(37, '127.0.0.1', '2023-10-12 06:15:07'),
(38, '127.0.0.1', '2023-10-12 06:15:33'),
(39, '127.0.0.1', '2023-10-12 06:15:37'),
(40, '127.0.0.1', '2023-10-12 06:15:38'),
(41, '127.0.0.1', '2023-10-12 06:15:56'),
(42, '127.0.0.1', '2023-10-12 06:16:00'),
(43, '127.0.0.1', '2023-10-12 06:22:01'),
(44, '127.0.0.1', '2023-10-12 06:22:28'),
(45, '127.0.0.1', '2023-10-12 06:31:04'),
(46, '127.0.0.1', '2023-10-12 06:41:16'),
(47, '127.0.0.1', '2023-10-12 06:41:22'),
(48, '127.0.0.1', '2023-10-12 06:41:23'),
(49, '127.0.0.1', '2023-10-12 06:41:24'),
(50, '127.0.0.1', '2023-10-12 06:41:24'),
(51, '127.0.0.1', '2023-10-12 06:41:24'),
(52, '127.0.0.1', '2023-10-12 06:41:24'),
(53, '127.0.0.1', '2023-10-12 06:41:24'),
(54, '127.0.0.1', '2023-10-12 06:41:25'),
(55, '127.0.0.1', '2023-10-12 06:41:25'),
(56, '127.0.0.1', '2023-10-13 03:49:47'),
(57, '127.0.0.1', '2023-10-13 05:22:28'),
(58, '127.0.0.1', '2023-10-13 05:22:44'),
(59, '127.0.0.1', '2023-10-13 05:22:51'),
(60, '127.0.0.1', '2023-10-13 05:22:56'),
(61, '127.0.0.1', '2023-10-13 05:23:48'),
(62, '127.0.0.1', '2023-11-27 11:49:02'),
(63, '127.0.0.1', '2023-11-27 11:49:08');

-- --------------------------------------------------------

--
-- Структура таблиці `users`
--

CREATE TABLE `users` (
  `id` int NOT NULL,
  `login` varchar(100) CHARACTER SET utf8mb4 COLLATE utf8mb4_general_ci NOT NULL,
```

```
  `name` varchar(100) CHARACTER SET utf8mb4 COLLATE utf8mb4_general_ci NOT NULL,
  `lname` varchar(100) CHARACTER SET utf8mb4 COLLATE utf8mb4_general_ci NOT NULL,
  `password` varchar(100) CHARACTER SET utf8mb4 COLLATE utf8mb4_general_ci NOT NULL,
  `country` varchar(100) CHARACTER SET utf8mb4 COLLATE utf8mb4_general_ci NOT NULL,
  `role` varchar(500) CHARACTER SET utf8mb4 COLLATE utf8mb4_general_ci DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci;

--
-- Дамп даних таблиці `users`
--

INSERT INTO `users` (`id`, `login`, `name`, `lname`, `password`, `country`, `role`)
VALUES
(21, 'user1', 'user1', 'user1', 'VGV6SmhJN1FVQXNBeUFvSXVUTTc4QT09', 'Україна',
'user');

--
-- Індекси збережених таблиць
--

--
-- Індекси таблиці `bookingtable`
--
ALTER TABLE `bookingtable`
  ADD PRIMARY KEY (`bookingID`),
  ADD UNIQUE KEY `bookingID` (`bookingID`),
  ADD KEY `bookingID_2` (`bookingID`),
  ADD KEY `bookingID_3` (`bookingID`),
  ADD KEY `bookingID_4` (`bookingID`);

--
-- Індекси таблиці `feedbacktable`
--
ALTER TABLE `feedbacktable`
  ADD PRIMARY KEY (`id`),
  ADD UNIQUE KEY `msgID` (`id`);

--
-- Індекси таблиці `franchase`
--
ALTER TABLE `franchase`
  ADD PRIMARY KEY (`id`);

--
-- Індекси таблиці `movietable`
--
ALTER TABLE `movietable`
  ADD PRIMARY KEY (`movieID`),
  ADD UNIQUE KEY `movieID` (`movieID`);

--
-- Індекси таблиці `request_log`
--
ALTER TABLE `request_log`
  ADD PRIMARY KEY (`id`);

--
-- Індекси таблиці `users`
--
ALTER TABLE `users`
  ADD PRIMARY KEY (`id`);

--
```

```
-- AUTO_INCREMENT для збережених таблиць
--

--
-- AUTO_INCREMENT для таблиці `bookingtable`
--
ALTER TABLE `bookingtable`
  MODIFY `bookingID` int NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=43;

--
-- AUTO_INCREMENT для таблиці `feedbacktable`
--
ALTER TABLE `feedbacktable`
  MODIFY `id` int NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=6;

--
-- AUTO_INCREMENT для таблиці `franchase`
--
ALTER TABLE `franchase`
  MODIFY `id` int NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=3;

--
-- AUTO_INCREMENT для таблиці `movietable`
--
ALTER TABLE `movietable`
  MODIFY `movieID` int NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=17;

--
-- AUTO_INCREMENT для таблиці `request_log`
--
ALTER TABLE `request_log`
  MODIFY `id` int NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=64;

--
-- AUTO_INCREMENT для таблиці `users`
--
ALTER TABLE `users`
  MODIFY `id` int NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=22;
COMMIT;

/*!40101 SET CHARACTER_SET_CLIENT=@OLD_CHARACTER_SET_CLIENT */;
/*!40101 SET CHARACTER_SET_RESULTS=@OLD_CHARACTER_SET_RESULTS */;
/*!40101 SET COLLATION_CONNECTION=@OLD_COLLATION_CONNECTION */;
```

ДОДАТОК Б
Програмний код

Index.php

```
<?php
session_start();
if($_SESSION['itlogin'] == ""){
    header('Location: /auth.php');
}else{
    $login = $_SESSION['itlogin'];
}

?>

<!DOCTYPE html>
<html lang="en">
<?php
ini_set('display_errors', 0);
ini_set('display_startup_errors', 0);
error_reporting(E_ALL);
?>
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <link rel="stylesheet" href="style/styles.css">
    <link rel="stylesheet"
href="https://use.fontawesome.com/releases/v5.7.2/css/all.css"
        integrity="sha384-
fnmOCqbTlWIlj8LyTjo7mOUStjsKC4pOpQbqyi7RrhN7udi9RwhKkMHpvLbHG9Sr"
crossorigin="anonymous">
    <title>Diploma AZS</title>
    <link rel="icon" type="image/png" href="img/logo.png">
</head>

<body>
    <?php
    $link = mysqli_connect("localhost", "root", "", "azs");
    $sql = "SELECT * FROM movietable";    ?>
    <header></header>
    <div id="home-section-1" class="movie-show-container">
        <h1>Наші послуги</h1>
        <h3>Оберіть паливно - мастильні матеріали, які ви хочете придбати</h3>

        <div class="movies-container">

            <?php
                        if($result = mysqli_query($link, $sql)){
                            $lines = mysqli_num_rows($result);
                            if(mysqli_num_rows($result) > 0){
                                for ($i = 0; $i <= $lines - 1; $i++){
                                    $row = mysqli_fetch_array($result);
                                    echo '<div class="movie-box">';
                                    echo '<img src="'. $row['movieImg'] .'" alt="
">';
                                    echo '<div class="movie-info ">';
                                    echo '<h3>'. $row['movieTitle'] .'</h3>';
```

```
                                    echo '<a
href="booking.php?id='.$row['movieID'].'"><i class="fas fa-ticket-alt"></i> Придбати
</a>';
                                    echo '</div>';
                                    echo '</div>';
                                }
                                mysqli_free_result($result);
                            } else{
                                echo '<h4 class="no-annot">На даний момент немає
вільних лікарів</h4>';
                            }                           } else{
                            echo "ERROR: Could not able to execute $sql. " .
mysqli_error($link);
                        }

                        // Close connection
                        mysqli_close($link);
                        ?>
        </div>
    </div>
    <div id="home-section-2" class="services-section">
        <h1>Як користуватися сервісом ?</h1>
        <h3>3 кроки для обслуговування вашого авто!</h3>

        <div class="services-container">
            <div class="service-item">
                <div class="service-item-icon">
                    <i class="fas fa-4x fa-video"></i>
                </div>
                <h2>1. Оберіть необхідну вам послугу</h2>
                <p>Оберіть необхідний паливно-мастильинй матеріал</p>
            </div>
            <div class="service-item">
                <div class="service-item-icon">
                    <i class="fas fa-4x fa-credit-card"></i>
                </div>
                <h2>2. Заповніть форму на придбання</h2>
                <p>Введіть ваші особові дані щодо придбання</p>
            </div>
            <div class="service-item">
                <div class="service-item-icon">
                    <i class="fas fa-4x fa-theater-masks"></i>
                </div>
                <h2>3. Насолоджуйтесь сервісом</h2>
                <p>Наші співробітники самі виконають всі процеси</p>
            </div>
            <div class="service-item"></div>

        </div>
    </div>
    <div id="home-section-3" class="trailers-section">
        <h1 class="section-title"> Не знаєте який тип паливно-мастильного матеріалу
обрати ?</h1>
        <h3>Ознайомтеся з відео - прев'ю нашої продукції !</h3>
        <div class="trailers-grid">
            <div class="trailers-grid-item">
                <img src="img/1.png" alt="">
                <div class="trailer-item-info" data-video="6c-OE-mZnZw">
                    <h3>Бензин чи дизель ? Що обрати</h3>
                    <i class="far fa-3x fa-play-circle"></i>
                </div>
            </div>
            <div class="trailers-grid-item">
```

```
                <img src="img/2.png" alt="">
                <div class="trailer-item-info" data-video="0zmrsqvRqQo">
                    <h3>Плюси та мінуси бензинового двигуна</h3>
                    <i class="far fa-3x fa-play-circle"></i>
                </div>
            </div>
            <div class="trailers-grid-item">
                <img src="img/3.png" alt="">
                <div class="trailer-item-info" data-video="WZNwN4uFzQc">
                    <h3>Скільки можна зберігати бензин у каністрі</h3>
                    <i class="far fa-3x fa-play-circle"></i>
                </div>
            </div>
            <div class="trailers-grid-item">
                <img src="img/4.png" alt="">
                <div class="trailer-item-info" data-video="6EJq7Y7gHN8">
                    <h3>Вибір кращого бензина</h3>
                    <i class="far fa-3x fa-play-circle"></i>
                </div>
            </div>
            <div class="trailers-grid-item">
                <img src="img/5.png" alt="">
                <div class="trailer-item-info" data-video="2NjtNZawP3c">
                    <h3>Чому здешевшав бензин</h3>
                    <i class="far fa-3x fa-play-circle"></i>
                </div>
            </div>
            <div class="trailers-grid-item">
                <img src="img/6.png" alt="">
                <div class="trailer-item-info" data-video="kWB8y-ouwsU">
                    <h3>АИ92-АИ100</h3>
                    <i class="far fa-3x fa-play-circle"></i>
                </div>
            </div>
        </div>
    </div>
    <footer></footer>

    <script src="scripts/jquery-3.3.1.min.js "></script>
    <script src="scripts/script.js "></script>
</body>

</html>
```

Auth.php

```
<?php
session_start();

?>



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

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <link rel="stylesheet" href="style/styles.css">
```

```
    <link rel="stylesheet"
href="https://use.fontawesome.com/releases/v5.7.2/css/all.css" integrity="sha384-
fnmOCqbTlWIlj8LyTjo7mOUStjsKC4pOpQbqyi7RrhN7udi9RwhKkMHpvLbHG9Sr"
crossorigin="anonymous">
    <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/css/bootstrap.min.css"
rel="stylesheet" integrity="sha384-
EVSTQN3/azprG1Anm3QDgpJLIm9Nao0Yz1ztcQTwFspd3yD65VohhpuuCOmLASjC"
crossorigin="anonymous">
    <link href="https://cdn.jsdelivr.net/npm/select2@4.1.0-
rc.0/dist/css/select2.min.css" rel="stylesheet" />
    <title>Авторизація</title>
    <link rel="icon" type="image/png" href="img/logo.png">
</head>

<body>
<?php
        $connection = mysqli_connect("localhost", "root", "", "azs");
?>
<div class="navbar-brand">
    <a href="index.php">
        <h1 class="navbar-heading">Diploma AZS</h1>
    </a>
</div>
<div>
<div mt="150px" class="container">
    <div class="row" style="margin-bottom: 20%;">
        <div class="col-3"></div>
        <div class="col-6">
            <h1><center>Будь ласка, введіть свій логін та пароль</center></h1><br>

            <form action="" method="POST" >
                <div class="box-body">

                    <div class="step_1">
                        <div class="form-group">
                            <label for="First_name">Логін:</label>
                            <input type="text" class="form-control first_name "
name="name"  required>
                        </div><br>


                        <div class="form-group">
                            <label for="subject">Пароль:</label>
                            <input  class="form-control subject"  required
type="text"   name="position"   >
                        </div><br>


                        <div class="box-footer">
                            <button type="submit" style="width: 100%;" name="submit"
value="submit" class="btn btn-danger">Увійти</button>
                            <button type="button" style="width: 100%; margin-top: 2%;
background-color: #1b9b4c;" onclick="window.location.href = 'registration.php';"
class="btn btn-danger">Зареєструватися</button>
                        </div>
                    </div>
                </div>
                <?php
                function encrypt_decrypt($string, $action = 'encrypt')
                {
                    $encrypt_method = "AES-256-CBC";
                    $secret_key = 'ihoi2h632h6j24h623jk';
```

```
                    $secret_iv = 'ihoi2h632h6j24h623jk';
                    $key = hash('sha256', $secret_key);
                    $iv = substr(hash('sha256', $secret_iv), 0, 16); // sha256 is
hash_hmac_algo
                    if ($action == 'encrypt') {
                        $output = openssl_encrypt($string, $encrypt_method, $key, 0,
$iv);
                        $output = base64_encode($output);
                    } else if ($action == 'decrypt') {
                        $output = openssl_decrypt(base64_decode($string),
$encrypt_method, $key, 0, $iv);
                    }
                    return $output;
                }
                if (isset($_POST['submit'])) {
                    $encryptedpass = encrypt_decrypt($_POST['position'],'encrypt');
                    // Екранування вхідних даних
                    $tyy = mysqli_real_escape_string($connection, $_POST['name']);
                    $pass = mysqli_real_escape_string($connection, $encryptedpass);

                    // Підготовлений запит для вибору ID користувача за логіном
                    $sql = "SELECT id FROM users WHERE login = ?";
                    $stmt = mysqli_prepare($connection, $sql);
                    mysqli_stmt_bind_param($stmt, "s", $tyy);
                    mysqli_stmt_execute($stmt);
                    $result = mysqli_stmt_get_result($stmt);

                    // Підготовлений запит для перевірки логіна та пароля
                    $check = "SELECT id FROM users WHERE login = ? and password = ?";
                    $stmt = mysqli_prepare($connection, $check);
                    mysqli_stmt_bind_param($stmt, "ss", $tyy, $pass);
                    mysqli_stmt_execute($stmt);
                    $res = mysqli_stmt_get_result($stmt);

                    if (mysqli_num_rows($result) == 0) {
                        echo "Такого користувача не існує, будь ласка,
зареєструйтесь!";
                    } elseif (mysqli_num_rows($res) == 0) {
                        echo "<script>alert('Введений пароль невірний, спробуйте
знову')</script>";

                    } else {
                        echo "<script>alert('Успішна авторизація')</script>";
                        $_SESSION['itlogin'] = $tyy;
                        header("location: /index.php");
                    }
                }
                // Перевірка кількості запитів від одного IP-адреси
                function checkRequestLimit($ip, $limit, $interval) {
                    $connection = mysqli_connect("localhost", "root", "", "azs");
                    // Вибрати кількість запитів за останній інтервал від цього IP
                    $query = "SELECT COUNT(*) as request_count FROM request_log WHERE
ip = ? AND created_at >= NOW() - INTERVAL ? SECOND";
                    $stmt = mysqli_prepare($connection, $query);
                    mysqli_stmt_bind_param($stmt, "si", $ip, $interval);
                    mysqli_stmt_execute($stmt);
                    $result = mysqli_stmt_get_result($stmt);
                    $row = mysqli_fetch_assoc($result);

                    // Перевірити, чи перевищено ліміт запитів
                    if ($row["request_count"] >= $limit) {
                        return false; // Перевищено ліміт запитів
                    }
```

```
                    // Якщо не перевищено ліміт, зберегти запит в лог
                    $insertQuery = "INSERT INTO request_log (ip) VALUES (?)";
                    $stmt = mysqli_prepare($connection, $insertQuery);
                    mysqli_stmt_bind_param($stmt, "s", $ip);
                    mysqli_stmt_execute($stmt);

                    return true; // Запит допущений
                }

                // Перевірка запиту перед виконанням
                $ip = $_SERVER['REMOTE_ADDR'];
                $limit = 10; // Максимальна кількість запитів
                $interval = 60; // Інтервал в секундах (60 секунд)

                if (checkRequestLimit($ip, $limit, $interval)) {
                    // Виконати запит
                    // ...
                } else {
                    // Повідомлення про перевищення ліміту запитів
                    echo "<script>alert('Ви перевищили ліміт запитів. Спробуйте
пізніше.')</script>";
                }

                ?>
            </form>
        </div>
    </div>
</div>
</div>
<footer></footer>
    <script src="scripts/jquery-3.3.1.min.js "></script>
    <script src="scripts/owl.carousel.min.js "></script>
    <script src="scripts/script.js "></script>
</body>

</html>
```

Registration.php

```
<?php
session_start();
?>

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

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <link rel="stylesheet" href="style/styles.css">
    <link rel="stylesheet"
href="https://use.fontawesome.com/releases/v5.7.2/css/all.css" integrity="sha384-
fnmOCqbTlWIlj8LyTjo7mOUStjsKC4pOpQbqyi7RrhN7udi9RwhKkMHpvLbHG9Sr"
crossorigin="anonymous">
    <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/css/bootstrap.min.css"
rel="stylesheet" integrity="sha384-
EVSTQN3/azprG1Anm3QDgpJLIm9Nao0Yz1ztcQTwFspd3yD65VohhpuuCOmLASjC"
crossorigin="anonymous">
```

```
    <link href="https://cdn.jsdelivr.net/npm/select2@4.1.0-
rc.0/dist/css/select2.min.css" rel="stylesheet" />
    <title>Реєстрація</title>
    <link rel="icon" type="image/png" href="img/logo.png">
</head>

<body>
<?php
        $link = mysqli_connect("localhost", "root", "", "azs");
?>
<div class="navbar-brand">
    <a href="index.php">
        <h1 class="navbar-heading">Diploma AZS</h1>
    </a>
</div>
<div>
<div mt="150px" class="container">
    <div class="row">
        <div class="col-3"></div>
        <div class="col-6">
            <h1><center>Будьласка, заповніть форму для реєстрації на
порталі</center></h1><br>

            <form action="" method="POST" >
                <div class="box-body">

                    <div class="step_1">
                        <div class="form-group">
                            <label for="First_name">Логін:</label>
                            <input type="text" class="form-control first_name "
name="name"  required>
                        </div><br>

                        <div class="form-group">
                            <label for="Second_name">Ім'я:</label>
                            <input type="text" class="form-control second_name"
name="sname"   required>
                        </div><br>

                        <div class="form-group">
                            <label for="birthday">Прізвище:</label>
                            <input  class="form-control birthday"  required
type="text"   name="company"   >
                        </div><br>

                        <div class="form-group">
                            <label for="subject">Пароль:</label>
                            <input  class="form-control subject"  required
type="text"   name="position"   >
                        </div><br>

                        <div class="form-group">
                            <label for="country">Країна:</label>
                            <select class="form-control country" name="country"
id="country"  >
                                <?php
                                $arr = array("Україна","Польща","Казахстан", "Інші");
                                foreach ($arr as &$value) {
                                    echo "<option>".$value."</option>";
                                }?>
                            </select>
                        </div><br>
```

```
                        <div class="box-footer">
                            <button type="submit" style=" width: 100%;" name="submit"
value="submit" class="btn btn-danger">Зареєструватися</button>
                        </div>
                        <div class="auth_button">
                            <h4 style="margin-top: 5%;">Вже зареєстровані?</h4>
                            <a href="auth.php">Увійти</a>
                        </div>
                    </div>
                </div>
                <?php
                function encrypt_decrypt($string, $action = 'encrypt')
                {
                    $encrypt_method = "AES-256-CBC";
                    $secret_key = 'ihoi2h632h6j24h623jk';
                    $secret_iv = 'ihoi2h632h6j24h623jk';
                    $key = hash('sha256', $secret_key);
                    $iv = substr(hash('sha256', $secret_iv), 0, 16); // sha256 is
hash_hmac_algo
                    if ($action == 'encrypt') {
                        $output = openssl_encrypt($string, $encrypt_method, $key, 0,
$iv);
                        $output = base64_encode($output);
                    } else if ($action == 'decrypt') {
                        $output = openssl_decrypt(base64_decode($string),
$encrypt_method, $key, 0, $iv);
                    }
                    return $output;
                }
                if(isset($_POST['submit'])){
                    $tyy = $_POST['name'];
                    $sql = "SELECT id FROM users WHERE login = '$tyy'";
                    $result = mysqli_query($link,$sql);
                    if(mysqli_num_rows($result)==0){
                        $validpass = encrypt_decrypt($_POST["position"]);
                        $insert_query = "INSERT INTO
                            users ( login,
                                                name,
                                                lname,
                                                password,
                                                country,
                                                role
                                                )
                            VALUES (            '".$_POST["name"]."',
                                                '".$_POST["sname"]."',
                                                '".$_POST["company"]."',
                                                '$validpass',
                                                '".$_POST["country"]."',
                                                'user'
                                                )";
                        mysqli_query($link,$insert_query);
                        $_SESSION['itlogin'] = $tyy;
                        echo "<script>alert(\"Успішна регістрація!\");</script>";
                        echo '<script>  setTimeout(function () {
window.location.href = "index.php"; }); </script>';
                     }else{
                    echo "<script>alert(\"Цей логін вже зайнятий!\");</script>";
                     }                }
                ?>
            </form>
        </div>
    </div>
</div>
```

```
</div>
<footer></footer>
    <script src="scripts/jquery-3.3.1.min.js "></script>
    <script src="scripts/owl.carousel.min.js "></script>
    <script src="scripts/script.js "></script>
</body>

</html>
```

Booking.php

```
<!DOCTYPE html>
<html lang="en">
<?php
        $id = $_GET['id'];
        $link = mysqli_connect("localhost", "root", "", "azs");
$movieQuery = "SELECT * FROM movietable WHERE movieID = $id";
        $movieImageById =  mysqli_query($link,$movieQuery);
        $row =  mysqli_fetch_array($movieImageById)
?>

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <link rel="stylesheet" href="style/styles.css">
    <link rel="stylesheet"
href="https://use.fontawesome.com/releases/v5.7.2/css/all.css"
        integrity="sha384-
fnmOCqbTlWIlj8LyTjo7mOUStjsKC4pOpQbqyi7RrhN7udi9RwhKkMHpvLbHG9Sr"
crossorigin="anonymous">
    <title>Придбання товару <?php echo $row['movieTitle']; ?></title>
    <link rel="icon" type="image/png" href="img/logo.png">
</head>

<body style="background-color:#6e5a11;">
    <div class="booking-panel">
        <div class="booking-panel-section booking-panel-section1">
            <h1>Будь ласка, заповніть форму для придбання обраного товару!</h1>
        </div>
        <div class="booking-panel-section booking-panel-section2"
onclick="window.location.href='index.php'">
            <i class="fas fa-2x fa-times"></i>
        </div>
        <div class="booking-panel-section booking-panel-section3">
            <div class="movie-box">
                <?php
                    echo '<img src="'.$row['movieImg'].'" alt="">';
                    ?>
            </div>
        </div>
        <div class="booking-panel-section booking-panel-section4">
            <div class="title"><?php echo $row['movieTitle']; ?></div>
            <div class="movie-information">
                <table>

                    <tr>
                        <td>ЦІНА ЗА ЛІТР</td>
                        <td><?php echo $row['movieGenre']; ?></td>
                    </tr>
                    <tr>
```

```
                        <td>КІЛЬКІСТЬ У НАЯВНОСТІ</td>
                        <td><?php echo $row['movieDuration']; ?></td>
                    </tr>
                    <tr>
                        <td>СТУПІНЬ ЗТИСНЕННЯ</td>
                        <td><?php echo $row['movieRelDate']; ?></td>
                    </tr>
                    <tr>
                        <td>ВИРОБНИК</td>
                        <td><?php echo $row['movieDirector']; ?></td>
                    </tr>
                    <tr>
                        <td>КОРОТКИЙ ОПИС</td>
                        <td><?php echo $row['movieActors']; ?></td>
                    </tr>
                </table>
            </div>
            <div class="booking-form-container">
                <form action="" method="POST">



                    <input placeholder="Ваше ім'я" type="text" name="fName" required>

                    <input placeholder="Ваше прізвище" type="text" name="lName">

                    <input placeholder="Ваш номер телефону" type="text"
name="pNumber" required>


                    <select name="station" required>
                        <option value="" disabled selected>Номер станції</option>
                        <option>1</option>
                        <option>2</option>
                        <option>3</option>
                        <option>4</option>
                        <option>5</option>
                        <option>6</option>
                    </select>

                    <select name="column" required>
                        <option value="" disabled selected>Номер колонки</option>
                        <option>Колонка 1</option>
                        <option>Колонка 2</option>
                        <option>Колонка 3</option>
                        <option>Колонка 4</option>
                        <option>Колонка 5</option>
                        <option>Колонка 6</option>
                    </select>

                    <input placeholder="Кількість літрів" type="text" name="amount"
required>

                    <button type="submit" value="submit" name="submit" class="form-
btn">Залишити заявку</button>
                    <?php
                    $fNameErr = $pNumberErr= "";
                    $fName = $pNumber = "";

                    if(isset($_POST['submit'])){

                        // Перевірка наявності підключення
                        if ($link === false) {
```

```
                            die("Помилка підключення: " . mysqli_connect_error());
                        }

                        // Підготовка SQL-запиту з параметрами
                        $insert_query = "INSERT INTO
                    bookingTable (ServiceName, Name, Lname, Phone, Station, Col,
Amount)
                    VALUES (?, ?, ?, ?, ?, ?, ?)";

                        // Підготовка запиту і зв'язування параметрів
                        $stmt = mysqli_prepare($link, $insert_query);

                        if ($stmt) {
                            $serviceName = $row['movieTitle'];
                            $fName = $_POST["fName"];
                            $lName = $_POST["lName"];
                            $pNumber = $_POST["pNumber"];
                            $station = $_POST["station"];
                            $col = $_POST["column"];
                            $amount = $_POST["amount"];

                            mysqli_stmt_bind_param($stmt, "sssssss", $serviceName,
$fName, $lName, $pNumber, $station, $col, $amount);

                            // Виконання запиту
                            if (mysqli_stmt_execute($stmt)) {
                                echo "Запис успішно додано до бази даних.";
                            } else {
                                echo "Помилка виконання запиту: " .
mysqli_error($link);
                            }

                            // Закриття підготовленого запиту і роз'єднання з базою
даних
                            mysqli_stmt_close($stmt);
                            mysqli_close($link);
                        } else {
                            echo "Помилка підготовки запиту: " . mysqli_error($link);
                        }
                    }
                    ?>
                </form>
            </div>
        </div>
    </div>

    <script src="scripts/jquery-3.3.1.min.js "></script>
    <script src="scripts/script.js "></script>
</body>

</html>
```

Admin.php

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

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
```

```
    <title>Admin Dashboard</title>
    <link rel="icon" type="image/png" href="../img/logo.png">
    <link rel="stylesheet" href="../style/styles.css">
    <link rel="stylesheet"
href="https://use.fontawesome.com/releases/v5.7.2/css/all.css"
        integrity="sha384-
fnmOCqbTlWIlj8LyTjo7mOUStjsKC4pOpQbqyi7RrhN7udi9RwhKkMHpvLbHG9Sr"
crossorigin="anonymous">
</head>

<body>
    <?php
    $link = mysqli_connect("localhost", "root", "", "azs");
    $sql = "SELECT * FROM bookingTable";
    $bookingsNo=mysqli_num_rows(mysqli_query($link, $sql));
    $messagesNo=mysqli_num_rows(mysqli_query($link, "SELECT * FROM feedbackTable"));
    $moviesNo=mysqli_num_rows(mysqli_query($link, "SELECT * FROM movieTable"));
    $franchaseNo=mysqli_num_rows(mysqli_query($link, "SELECT * FROM franchase"));
    ?>
    <div class="admin-section-header">
        <div class="admin-logo">
            Diploma AZS
        </div>
        <div class="admin-login-info">
            <i class="far fa-bell admin-notification-button"></i>
            <i class="far fa-comment-alt"></i>
            <a href="#">Welcome, Admin</a>
            <img class="admin-user-avatar" src="../img/avatar.png" alt="">
        </div>
    </div>
    <div class="admin-container">
        <div class="admin-section admin-section1 ">
            <ul>
                <a href="adminfeedbacks.php"> <li> <i class="fas fa-sliders-
h"></i>Feedbacks <i class="fas admin-dropdown fa-chevron-right"></i></li></a>
                <li><div class="schedule-item schedule-item-selected"><i class="fas
fa-ticket-alt"></i>Bookings <i ></i></div></li>
                <a href="adminfranchises.php"><li class="admin-navigation-
schedule"><i class="fas fa-calendar-alt"></i>Franchases<i
                        class="fas admin-dropdown fa-chevron-right"></i>
                </li></a>

                <a href="adminservices.php"><li><i class="fas fa-film"></i>Services
<i class="fas admin-dropdown fa-chevron-right"></i></li></a>
            </ul>
        </div>
        <div class="admin-section admin-section2">
            <div class="admin-section-column">
                <div class="admin-section-panel admin-section-stats">
                    <div class="admin-section-stats-panel">
                        <i class="fas fa-ticket-alt" style="background-color:
#cf4545"></i>
                        <h2 style="color: #cf4545"><?php echo $bookingsNo ?></h2>
                        <h3>Bookings</h3>
                    </div>
                    <div class="admin-section-stats-panel">
                        <i class="fas fa-film" style="background-color: #4547cf"></i>
                        <h2 style="color: #4547cf"><?php echo $moviesNo ?></h2>
                        <h3>Services</h3>
                    </div>
                    <div class="admin-section-stats-panel">
                        <i class="fas fa-ticket-alt" style="background-color:
#bb3c95"></i>
```

```
                        <h2 style="color: #bb3c95"><?php echo $franchaseNo ?></h2>
                        <h3>Franchases</h3>
                    </div>
                    <div class="admin-section-stats-panel" style="border: none">
                        <i class="fas fa-envelope" style="background-color:
#3cbb6c"></i>
                        <h2 style="color: #3cbb6c"><?php echo $messagesNo ?></h2>
                        <h3>Feedbacks</h3>
                    </div>
                </div>
                <div class="admin-section-panel admin-section-panel1">
                    <div class="admin-panel-section-header">
                        <h2>Bookings</h2>
                        <i class="fas fa-ticket-alt" style="background-color:
#cf4545"></i>
                    </div>
                    <div class="admin-panel-section-content">
                        <?php
                        if($result = mysqli_query($link, $sql)){
                            if(mysqli_num_rows($result) > 0){
                                while($row = mysqli_fetch_array($result)){
                                    echo "<div class=\"admin-panel-section-booking-
item\">\n";
                                    echo "                                <div
class=\"admin-panel-section-booking-response\">\n";
                                    echo "                                    <a
href='deleteBooking.php?id=".$row['bookingID']."'><i class=\"fas fa-check accept-
booking\" title=\"Reject booking\"></i></a>\n";
                                    echo "                                    <a
href='deleteBooking.php?id=".$row['bookingID']."'><i class=\"fas fa-times decline-
booking\" title=\"Reject booking\"></i></a>\n";
                                    echo "                                </div>\n";
                                    echo "                                <div
class=\"admin-panel-section-booking-info\">\n";
                                    echo "                                    <div>\n";
                                    echo "                                        <h3>".
$row['ServiceName'] ."</h3>\n";
                                    echo "                                        <i
class=\"fas fa-circle \"></i>\n";
                                    echo "                                    </div>\n";
                                    echo "                                    <div>\n";
                                    echo "                                        <h4>".
$row['Name'] ."</h4>\n";
                                    echo "                                        <i
class=\"fas fa-circle \"></i>\n";
                                    echo "                                        <h4>".
$row['Lname'] ."</h4>\n";
                                    echo "                                        <i
class=\"fas fa-circle \"></i>\n";
                                    echo "                                        <h4>".
$row['Phone'] ."</h4>\n";
                                    echo "                                        <i
class=\"fas fa-circle \"></i>\n";
                                    echo "
<h4>Заправна станція № ". $row['Station'] ."</h4>\n";
                                    echo "                                        <i
class=\"fas fa-circle \"></i>\n";
                                    echo "
<h4>Колонка № ". $row['Col'] ."</h4>\n";
                                    echo "                                    </div>\n";
                                    echo "                                </div>\n";
                                    echo "                            </div>";
                                }
```

```
                                mysqli_free_result($result);
                            } else{
                                echo '<h4 class="no-annot">No Bookings right
now</h4>';
                            }
                        } else{
                            echo "ERROR: Could not able to execute $sql. " .
mysqli_error($link);
                        }
                        ?>
                    </div>
                </div>
            </div>

        </div>
    </div>

    <script src="../scripts/jquery-3.3.1.min.js "></script>
    <script src="../scripts/owl.carousel.min.js "></script>
    <script src="../scripts/script.js "></script>
</body>

</html>
```

ДОДАТОК В
Діаграма взаємодії програмних модулів

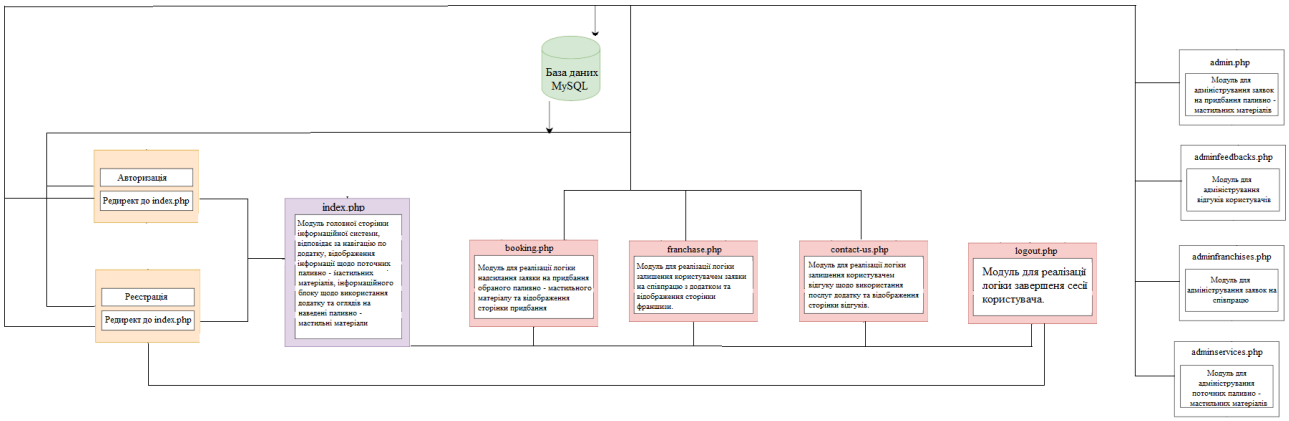


Рисунок В.1 – Діагарама взаємодії програмних модулів

ДОДАТОК Г
Тестування реєстрації користувача

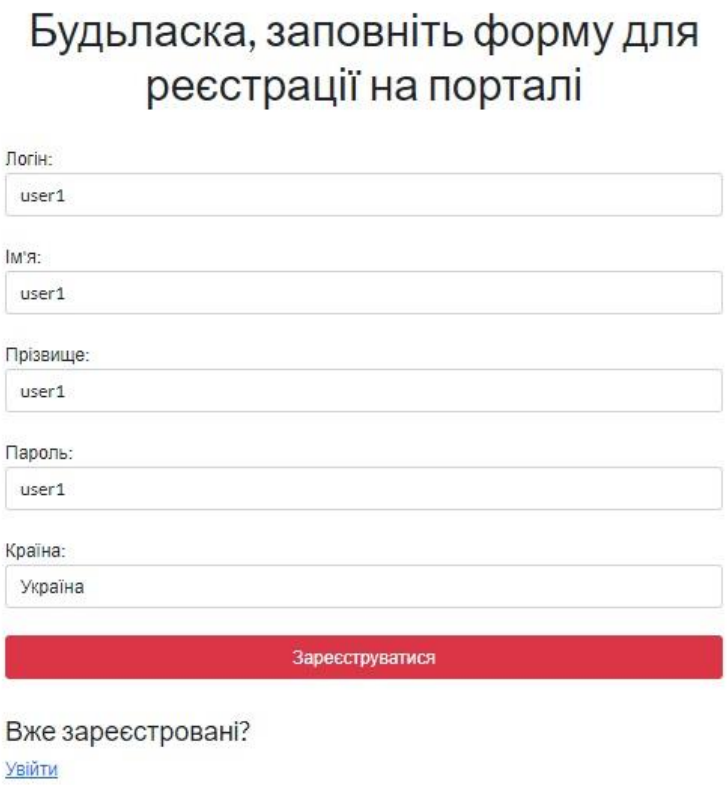


Рисунок Г.1 – Тестування системи

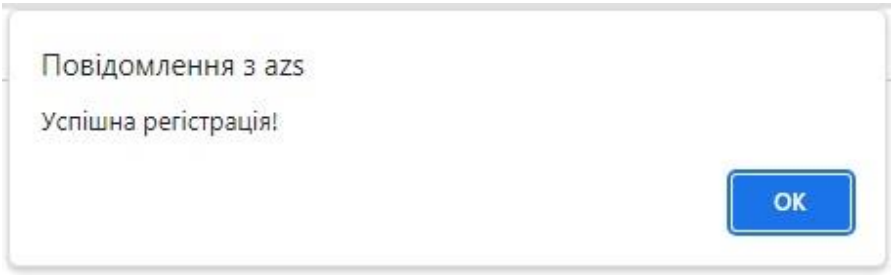


Рисунок Г.2 – Тестування системи

Рисунок Г.3 – Тестування системи

ДОДАТОК Д
Тестування авторизації користувача

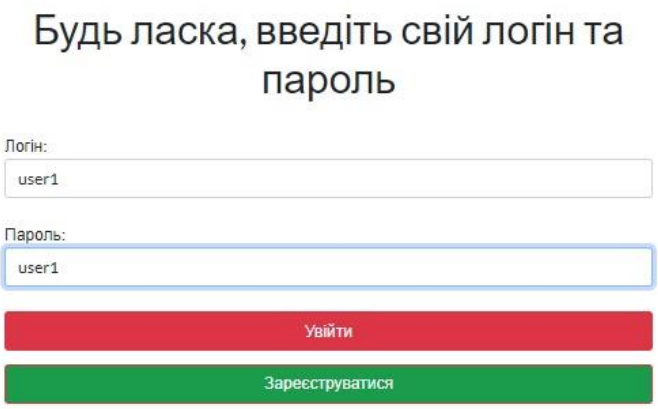


Рисунок Д.1 – Тестування системи

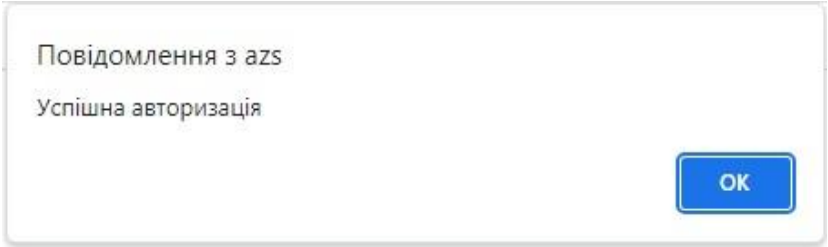


Рисунок Д.2 – Тестування системи

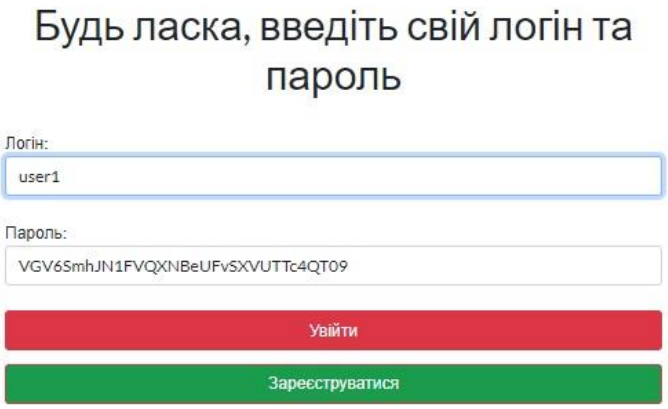


Рисунок Д.3 – Тестування системи

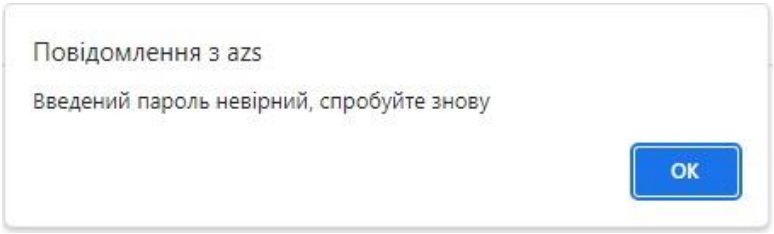


Рисунок Д.4 – Тестування системи

ДОДАТОК Е
Тестування впроваджених засобів безпеки проти SQL – ін’єкцій

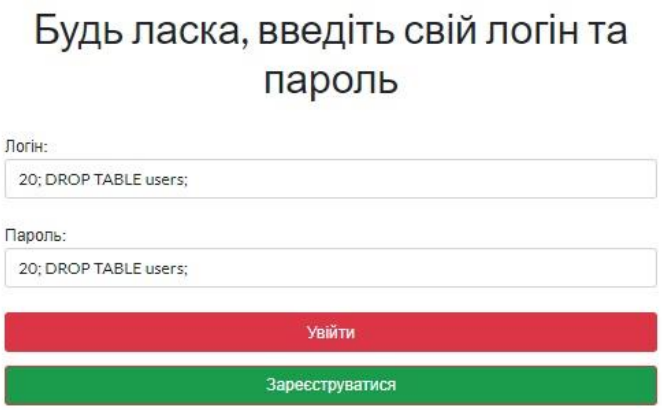
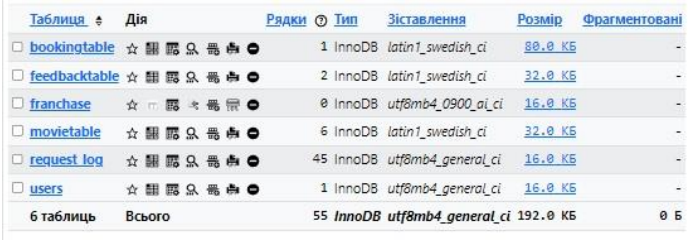
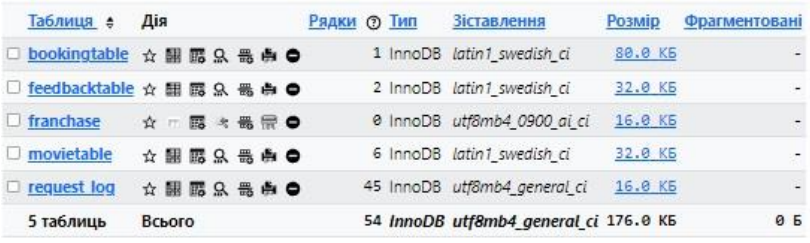


Рисунок Е.1 – Тестування системи



Таблиця	Дія	Рядки	Тип	Зіставлення	Розмір	Фрагментовані
bookingtable		1	InnoDB	latin1_swedish_ci	80.0 КБ	-
feedbacktable		2	InnoDB	latin1_swedish_ci	32.0 КБ	-
franchase		0	InnoDB	utf8mb4_0900_ai_ci	16.0 КБ	-
movietable		6	InnoDB	latin1_swedish_ci	32.0 КБ	-
request log		45	InnoDB	utf8mb4_general_ci	16.0 КБ	-
users		1	InnoDB	utf8mb4_general_ci	16.0 КБ	-
6 таблиць	Всього	55	InnoDB	utf8mb4_general_ci	192.0 КБ	0 Б

Рисунок Е.2 – Тестування системи



Таблиця	Дія	Рядки	Тип	Зіставлення	Розмір	Фрагментовані
bookingtable		1	InnoDB	latin1_swedish_ci	80.0 КБ	-
feedbacktable		2	InnoDB	latin1_swedish_ci	32.0 КБ	-
franchase		0	InnoDB	utf8mb4_0900_ai_ci	16.0 КБ	-
movietable		6	InnoDB	latin1_swedish_ci	32.0 КБ	-
request log		45	InnoDB	utf8mb4_general_ci	16.0 КБ	-
5 таблиць	Всього	54	InnoDB	utf8mb4_general_ci	176.0 КБ	0 Б

Рисунок Е.3 – Тестування системи

ДОДАТОК Ж
Тестування впровадженого алгоритму протидії DoS атакам

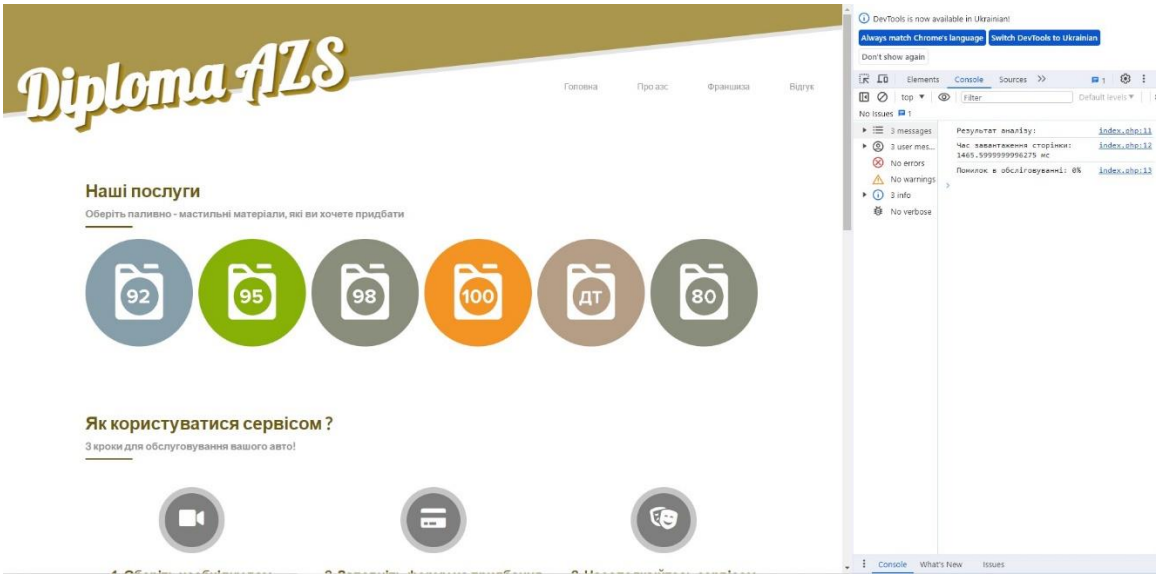


Рисунок Ж.1 – Тестування системи

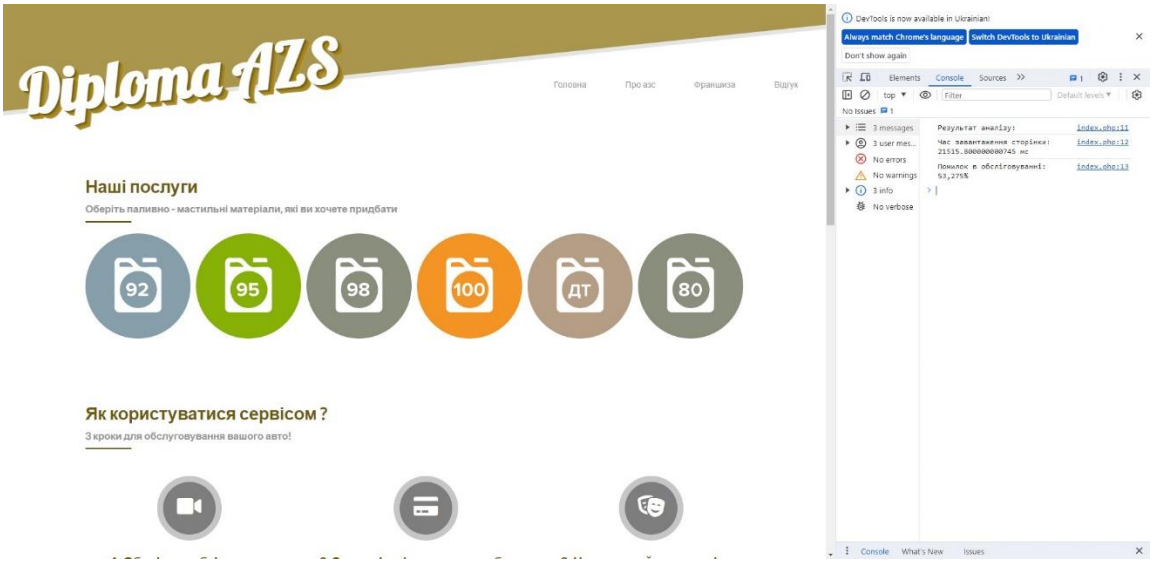


Рисунок Ж.2 – Тестування системи

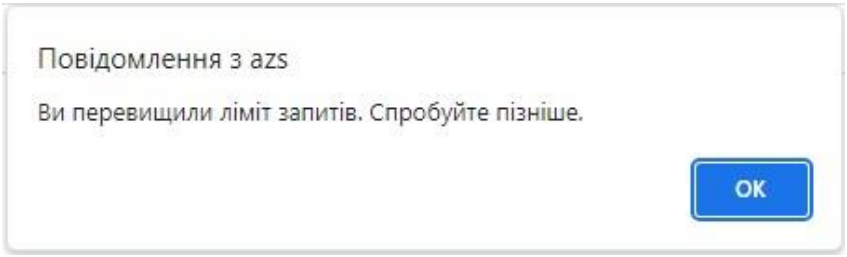


Рисунок Ж.3 – Тестування системи

ДОДАТОК З

Тестування процес автентифікації користувача з підвищеними правами доступу до адміністрування бази даних

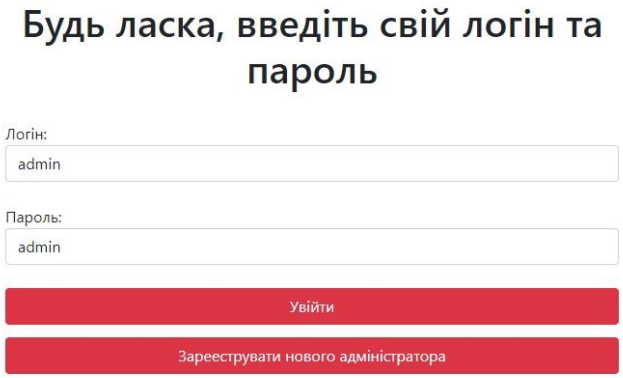


Рисунок З.1 – Тестування системи

Рисунок З.2 – Отриманий токен для автентифікації з підвищеними правами доступу до бази даних

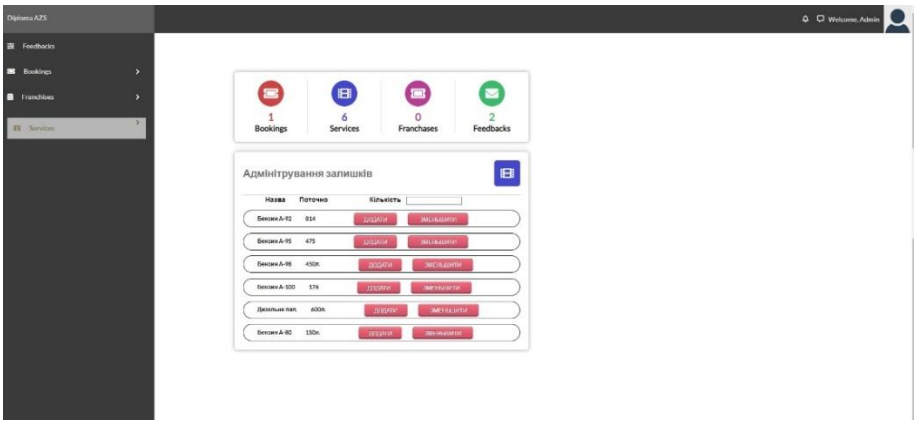


Рисунок 3.3 – Результат успішного отримання доступу до інтерфейсу користувача з підвищеними правами на адміністрування бази даних

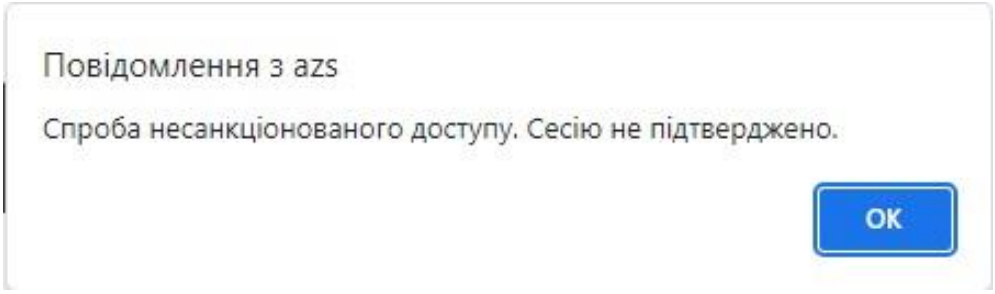


Рисунок 3.4 – Результат відмови в отриманні доступу до інтерфейсу користувача з підвищеними правами на адміністрування бази даних