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

**МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА**

на тему:

# «ІНФОРМАЦІЙНА ТЕХНОЛОГІЯ ВИЯВЛЕННЯ SQL–ІН'ЄКЦІЙ МЕТОДАМИ МАШИННОГО НАВЧАННЯ»

Виконав: студент 2 курсу, групи 1БС–22м
спеціальності 125 Кібербезпека
_____ Дарина ТІЩЕНКО
Керівник: к. т. н., доцент каф. ЗІ
_____ Леонід КУПЕРШТЕЙН
«12» 12 2023 р.
Опонент: к. т. н. доцент каф. ПЗ
_____ Олена КОВАЛЕНКО
«13» 12 2023 р.

**Допущено до захисту**

Завідувач кафедри ЗІ
д. т. н., проф.
_____ Володимир ЛУЖЕЦЬКИЙ
«14» 12 2023 р.

Вінниця ВНТУ – 2023 року

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

ЗАТВЕРДЖУЮ
Зав. кафедри ЗІ, д. т. н., проф.
Володимир ЛУЖЕЦЬКИЙ
19.09 2023 року

**ЗАВДАННЯ**
**НА МАГІСТЕРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ**
Тіщенко Дарині Сергіївні

1. Тема роботи: «Інформаційна технологія виявлення SQL–ін'єкцій методами машинного навчання» керівник роботи: Куперштейн Леонід Михайлович, к. т. н., доцент, затверджені наказом ВНТУ від 18 вересня 2023р. №247.
2. Строк подання студентом роботи: 13 грудня 2023 року.
3. Вихідні дані до роботи:
   - засіб повинен запускатись в операційній системі Windows;
   - засіб повинен виявляти загрози SQL–ін'єкцій;
   - засіб повинен надавати достовірність виявлення SQL–ін'єкцій.
4. Зміст текстової частини: Вступ. 1. Аналіз предметної області. 2. Розробка інформаційної технології. 3. Програмна реалізація інформаційної технології. 4. Економічна частина. Висновки. Перелік використаних джерел. Додатки
5. Перелік графічного матеріалу.

Схема типів SQL–ін'єкцій (плакат, А4). Схема процесів інформаційної технології виявлення SQL–ін'єкцій методами машинного навчання (плакат, А4). Схема етапів інформаційної технології виявлення SQL–ін'єкцій методами машинного навчання (плакат, А4). Архітектура інформаційної технології виявлення SQL–ін'єкцій методами машинного навчання (плакат, А4). Структурна схема інформаційної технології виявлення SQL–ін'єкцій методами машинного навчання (плакат, А4) Схема валідаційного процессу (плакат,А4).

6. Консультанти розділів роботи

| Розділ | Прізвище, ініціали та посада консультанта | Підпис, дата | |
|---|---|---|---|
| | | завдання видав | завдання прийняв |
| 1 | Куперштейн Л. М., доц. кафедри ЗІ | 19.09 | 26.09 |
| 2 | Куперштейн Л. М., доц. кафедри ЗІ | 19.09 | 29.09 |
| 3 | Куперштейн Л. М., доц. кафедри ЗІ | 19.09 | 26.10 |
| 4 | Ратушняк О. Г., к.т.н., доц. каф ЕВПМ | 19.09 | 27.11 |

6. Дата видачі завдання 1 вересня 2023 року

## КАЛЕНДАРНИЙ ПЛАН

| № з/п | Назва етапів магістерської кваліфікаційної роботи | Строк виконання етапів роботи | Примітка |
|---|---|---|---|
| 1 | Аналіз завдання. Вступ | 01.09.2023 – 10.09.2023 | |
| 2 | Аналіз літературних джерел за напрямком магістерської кваліфікаційної роботи | 10.09.2023 – 15.09.2023 | |
| 3 | Науково–технічне обґрунтування | 16.09.2023 – 22.09.2023 | |
| 4 | Розробка технічного завдання | 23.09.2023 – 29.09.2023 | |
| 5 | Розробка рішень | 30.09.2023 – 12.10.2023 | |
| 6 | Практична реалізація, моделювання, експериментування, результати | 14.10.2023 – 10.11.2023 | |
| 7 | Розробка розділу економічного обґрунтування доцільності розробки | 11.11.23 – 17.11.2023 | |
| 8 | Аналіз виконання ТЗ, висновки | 18.11.2023 – 24.11.2023 | |
| 9 | Оформлення пояснювальної записки | 25.11.2023 – 30.11.2023 | |
| 10 | Попередній захист та доопрацювання МКР | 28.11.2023 – 01.12.2023 | |
| 11 | Перевірка магістерської роботи на наявність плагіату | 02.12.2023 – 10.12.2023 | |
| 12 | Представлення МКР до захисту | 11.12.2023 – 14.12.2023 | |
| 13 | Захист МКР | 14.12.2023 – 21.12.2023 | |

Студент _____ Дарина ТІЩЕНКО
(підпис)

Керівник роботи _____ Леонід КУПЕРШТЕЙН
(підпис)

# АНОТАЦІЯ

УДК 004.56.05

Тіщенко Д. С. Інформаційна технологія виявлення SQL–ін'єкцій методами машинного навчання. Магістерська кваліфікаційна робота зі спеціальності 125 – Кібербезпека, освітня програма – Безпека інформаційних і комунікаційних систем. Вінниця: ВНТУ, 2023. 87 с.

Укр. мовою. Бібліогр.: 28 назв; рис.: 46; табл.: 19.

Магістерська кваліфікаційна робота присвячена розробці інформаційної технології виявлення SQL–ін'єкцій методами машинного навчання. В рамках роботи проведено аналіз існуючих методів та засобів виявлення SQL–ін'єкцій. На основі існуючих методів було запропоновано оптимальний підхід до виявлення SQL–ін'єкцій з використанням моделі машинного навчання. Розроблено програмний засіб, який дозволив протестувати спроектовану технологію.

Ілюстративна частина складається з 6 плакатів з демонстрацією результатів моделювання і проведених досліджень.

В економічному розділі оцінено витрати на розробку технології та програмного засобу.

Ключові слова: SQL–ін'єкція, виявлення SQL–ін’єкцій, веб–додаток, машинне навчання, збір даних.

# ABSTRACT

Tishchenko D. S. Information technology for detection of SQL injections using machine learning methods. Master's thesis on specialty 125 – Cybersecurity, educational program – Security of information and communication systems. Vinnytsia: VNTU, 2023. 87 p.

In Ukrainian. Bibliographer: 28 titles; fig.: 46; tabl.: 19.

The master's thesis is devoted to the development of information technology for detecting SQL injections using machine learning methods. As part of the work, an analysis of existing methods and tools for detecting SQL injections was carried out. Based on existing methods, an optimal approach to detecting SQL injections using a machine learning model was proposed. A software tool was developed that allowed testing the designed technology.

The illustrative part consists of 6 posters with a demonstration of the results of modeling and conducted research.

The economic section estimates the costs of technology and software development.

Keywords: SQL injection, SQL injection detection, web application, machine learning, data collection.

# ЗМІСТ

# ВСТУП

Безпека пристроїв, систем і мереж є життєво важливою, оскільки світ постійно розвивається і залежить від автоматизованих складних систем. Дослідники розробляють різні методи і засоби виявлення аномалій, щоб призупинити та контролювати вплив загроз на системи. Однак існуючі рішення часто не працюють, коли мова йде про пристосування до постійно–змінної архітектури застосунків.

З кожним роком зростає кількість компаній, які використовують веб–технології для підвищення продуктивності та залучення нових клієнтів. Такі організації включають державні та місцеві органи влади, а також комерційні компанії різних форм власності. Інтернет–сервіси несуть з собою безліч переваг, але з ростом числа додатків збільшується і кількість кіберзагроз.

Ін’єкції є однією з найдавніших і найнебезпечніших загроз. Вони можуть завдати шкоди цілісності, конфіденційності та доступності даних.

Зловмисник вносить довільний код до програми під час ін'єкційної атаки. Інтерпретатор обробляє цей код як частину команди або запиту. Це може змінити логіку роботи програми та призвести до непоправних наслідків.

Зазвичай атаки типу SQL–ін’єкції стосуються веб–додатків, але будь–яка клієнт–серверна та серверна програма, яка працює з системами управління базами даних, є уразливою до даної атаки.

**Актуальність теми.** Актуальність цієї теми полягає в тому, що інформація є цінною, має матеріальне або нематеріальне вираження та потребує захисту від різноманітних негативних впливів. Одним із таких вплтвів є SQL–ін’єкція. Це пов’язано з тим, що вразливості баз даних легко знайти та використати, що призводить до їх використання зловмисниками. Таким чином, виявлення SQL–ін’єкцій за допомогою методів машинного навчання є важливим завданням для інформаційних технологій.

**Об’єктом дослідження** є процеси виявлення SQL–ін’єкцій.

**Предметом дослідження** методи та засоби виявлення SQL– ін’єкцій.

**Метою роботи** підвищення ефективності виявлення SQL–ін’єкцій за допомогою використання методів машинного навчання. Для досягнення мети необхідно виконати наступні завдання:

- проаналізувати існуючі методи та засоби виявлення SQL–ін’єкцій;
- розробити інформаційну технологію;
- виконати навчання моделі машинного навчання;
- розробити програмний засіб, що реалізує інформаційну технологію;
- виконати тестування програмного засобу.

**Методи дослідження.** Для реалізації поставлених задач були використані статистичні методи для підготовки вхідної та вихідної інформації для моделювання моделі машинного навчання, методи проектування програмного забезпечення для розробки та верифікації інтелектуальної технології, методи побудови моделей машинного навчання.

**Наукова новизна.** Вдосконалену інформаційну технологію виявлення SQL–ін’єкцій, яка за рахунок використання методів машинного навчання, а саме дерев рішень, дозволяє автоматично з великою достовірністю виявляти та запобігати загрозам.

**Практична цінність** полягає у тому, що натреновано інтелектуальну модель на основі дерев рішень для виявлення SQL–ін’єкцій, яку можна використовувати у будь–яких практичних кейсах.

Результати бакалаврської роботи доповідалося на міжнародній науково-практичній Інтернет-конференції студентів, аспірантів та молодих науковців «Молодь в науці: дослідження, проблеми, перспективи (МН-2024)» [1].

# 1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

## 1.1 Аналіз типів SQL–ін’єкцій

Розуміння поширених уразливостей веб–додатків допомагає краще підготуватися до захисту даних від таких атак. Дослідження дозволяють користувачам і розробникам бути краще підготовленими для боротьби з найпоширенішими атаками та приймати рішення, щоб запобігти майбутнім атакам на їхні додатки.

Атаки SQL–ін’єкції, також відомі як SQLIA, є однією з найбільш поширених загроз безпеці програм, керованих базами даних. OWASP (Open Web Application Security Project) щороку створює ранкінг кіберзагроз, зокрема найбільш поширених [2]. Список базується на реальному досвіду спеціалістів із кібербезпеки та інших IT–фахівців, а також на необроблених даних, зібраних за допомогою автоматизованого тестування. Як показано на рисунку 1.1, атаки типу ін’єкції, такі як SQL, NoSQL, OS і LDAP, були найпоширенішими кібератаками в 2017, 2018, 2020 та 2022 роках.

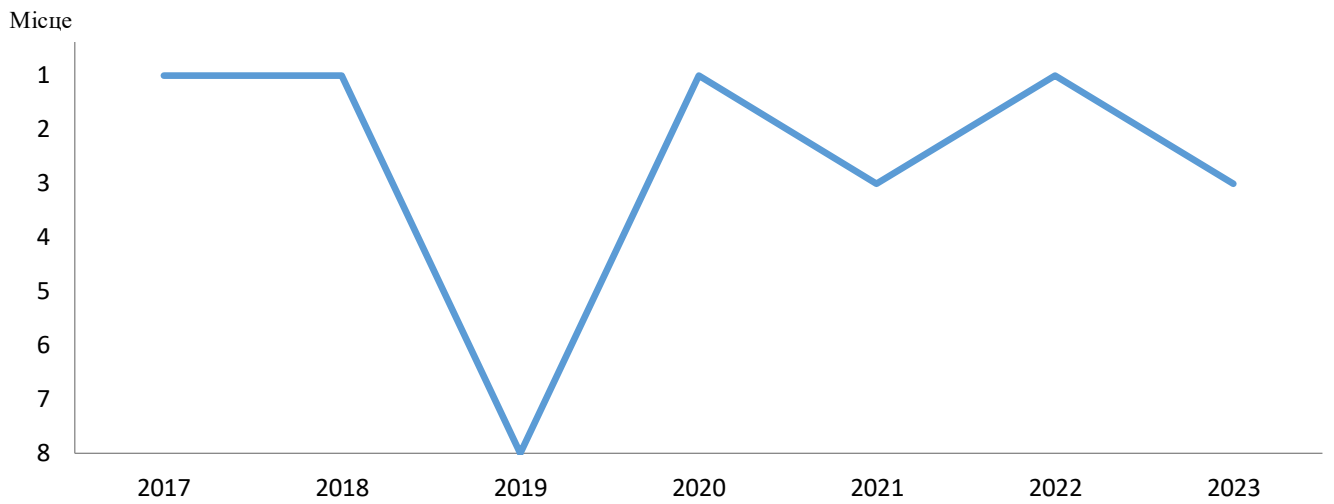



Рисунок 1.1 – Місце атак типу ін’єкції в рейтингу OWASP Top–10

Понад 40 000 атак відбувається щодня в усьому світі, згідно зі статистичними даними, і це значна проблема, яка вимагає вирішення для

багатьох веб–систем [3].

Системи зломів часто використовують технології введення коду для отримання інформації, розширення привілеїв або несанкціонованого доступу до системи, що порушує цілісність, доступність і конфіденційність даних. Типові результати впровадження SQL–ін'єкції наведено в таблиці 1.1.

Таблиця 1.1 Приклади загроз при впровадженні SQL–ін'єкції.

| Види загроз | Приклади впровадження SQL–ін'єкції |
|---|---|
| Підвищення привілеїв | Використання облікової інформації адміністратора |
| Розкриття інформації | Отримання інформації про дебетову або кредитну картку |
| Спуфінг | Використання даних іншого користувача |
| Відмова | Видалення журналів бази даних |
| Фальсифікація | Зміна даних в базі даних |
| Відмова в обслуговуванні | Виконання ресурсомістких запитів SQL |

Як показано на рисунку 1.2, пряма взаємодія між користувачем і базою даних є основною проблемою, та головною небезпекою.

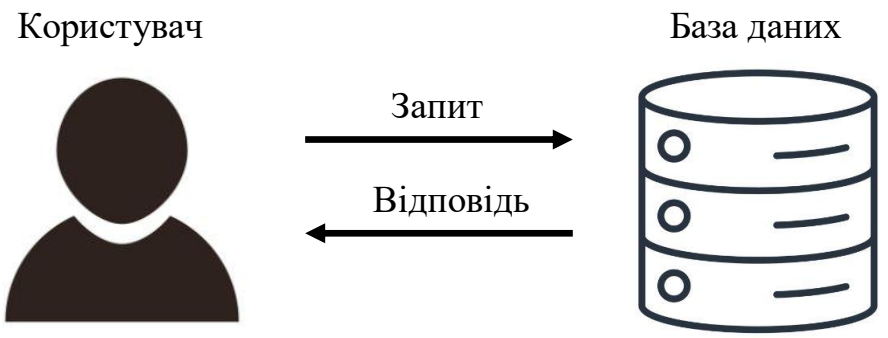



Рисунок 1.2 – Взаємодія користувача з базою даних

Атакуючий може вводити зловмисний код за допомогою SQL–запитів, змушуючи систему виконати непередбачувані запити до бази даних, що

дозволяє йому переглядати конфіденційні дані та навіть модифікувати їх (вставляти, оновлювати або видаляти) (рис. 1.3).

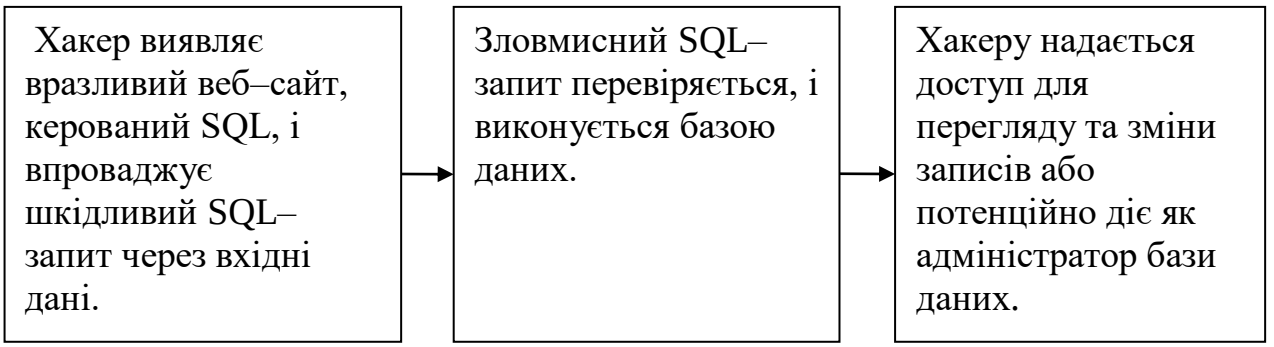



Рисунок 1.3 – Алгоритм вповадження SQL–ін’єкцій

Існує багато типів SQL–ін’єкцій. Зазвичай вони поділяються на три категорії: In–band SQLi, Inferential SQLi (Blind) і Out–of–band SQLi (рис.1.4) [3].

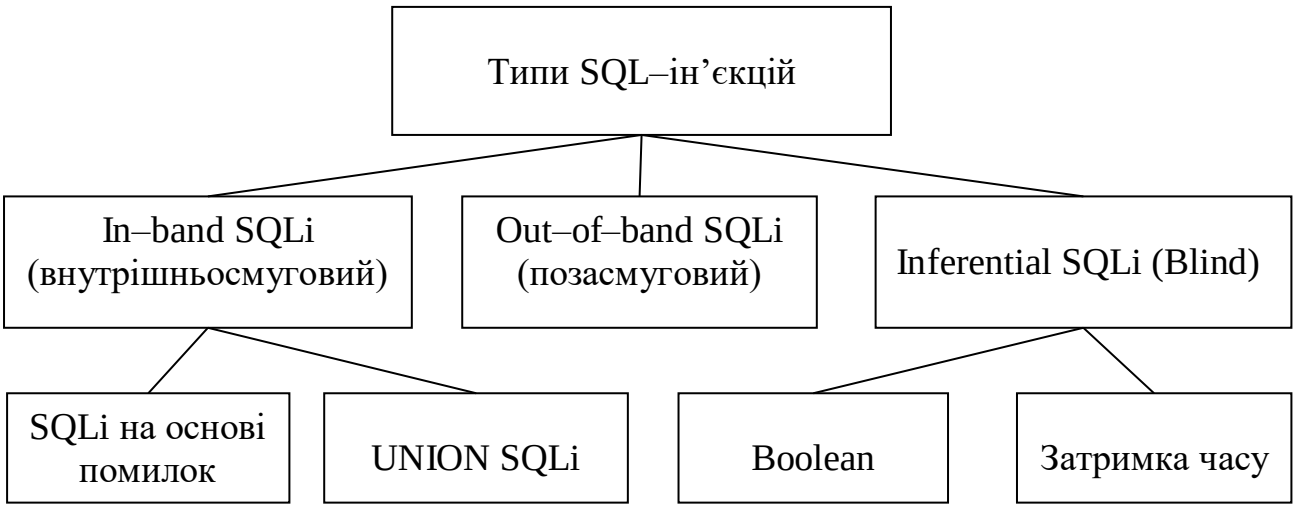



Рисунок 1.4 – Типи SQL–ін’єкцій

Розглянемо приклади найрозповсюдженіших SQL–ін’єкцій:

- In–band SQLi. Щоб розпочати свої атаки та отримати результати, зловмисник використовує той самий канал зв’язку. Внутрішньосмугові SQL–ін’єкції є одним із найпоширеніших типів атак через його простоту та ефективність [4]. Розглянемо два варіанти цієї стратегії:

- SQLi на основі помилок (Error–based SQLi) — зловмисник виконує дії, які змушують базу даних генерувати повідомлення про помилки. Дані, надані цими повідомленнями про помилки, можуть допомогти зловмиснику дізнатися про

структуру бази даних [4].

- UNION SQLi — цей метод використовує переваги оператора UNION SQL, який об’єднує кілька операторів вибору, створених базою даних, для отримання єдиної відповіді HTTP. У цій відповіді можуть бути дані, які можуть використати злочинці [4].

У більшості випадків частина запиту, з якою проводяться маніпуляції, міститься в умові оператора WHERE. Є можливість змінити запит так, щоб він повертав записи, які не були призначені запитом. Це можна зробити за допомогою оператора UNION, який надає можливість використовувати більше одного оператора SELECT в одному запиті [4]. Розглянемо наступний запит:

```
"SELECT Company FROM Shippers WHERE 1 = 1
UNION ALL SELECT CompanyFROM Customers WHERE 1 = 1"
```

Записи першого і другого запиту будуть повернуті цим запитом. Оператор ALL необхідний для того, щоб уникнути SELECT DISTINCT, тобто виведення тільки одного запиту.

В даному випадку параметр City укладений в круглі дужки, тому рядок ін'єкції також повинен їх утримувати. Змінимо попередню ін'єкцію наступним чином:

```
" ') UNION SELECT OtherField FROM OtherTable WHERE (' '='"
```

Таким чином, на сервер буде відправлено наступний запит, який буде правильним з точки зору синтаксису:

```
"SELECT Name, Surname, Title, FROM Employees WHERE (City = '')
    UNION SELECT OtherField From OtherTable WHERE ( '' = '')"
```

–Inferential SQLi (Blind) – зловмисник передає дані на сервер і спостерігає за його реакцією та поведінкою, щоб дізнатися про його структуру. Таким чином, зловмисник не може побачити інформацію про атаку в смузі. Оскільки дані не передаються зловмиснику з бази даних веб–сайту, цей метод

називається сліпим SQL–ін’єкціями [5].

Сліпі SQL–ін’єкції зазвичай виконуються повільніше, але можуть бути такими ж шкідливими, оскільки вони залежать від відповіді та моделей поведінки сервера. Сліпі SQL–ін’єкції можна класифікувати наступним чином:

–Логічний (Boolean) — зловмисник надсилає запит SQL до бази даних, що змушує програму повернути результат. Результат буде різним залежно від того, правдивий чи хибний запит. Інформація у відповіді HTTP може змінитися або залишатися незмінною залежно від результату. Після цього злочинець може визначити, чи повідомлення призвело до правильного чи неправильного результату [5].

–На основі часу — Зловмисник надсилає запит SQL до бази даних, що змушує базу даних чекати , перш ніж вона зможе відповісти. Зловмисник може визначити, чи є запит істинним чи хибним, використовуючи час, необхідний базі даних для відповіді. Відповідь HTTP буде створена або миттєво, або через час очікування. Зловмисник може визначити, чи є результат запиту TRUE або FALSE, навіть якщо дані з бази даних не повертаються. Оскільки зловмисник повинен перераховувати базу даних символ за символом, ця атака часто є повільною, це особливо стосується великих баз даних. [5]. Розглянемо наступний запит:

```
And SLEEP(10) if sleep then Vulnerable
OR SLEEP(10) if sleep then vulnerable
```

–Out–of–band SQLi (позасмугові). Зловмисник може здійснити таку атаку лише тоді, коли певні функції ввімкнено на сервері бази даних веб–додатка. Цей тип атаки часто використовується як заміна інференційним і внутрішньосмуговим методам SQL–ін’єкцій [6].

Позасмугові SQL–ін’єкції виконуються, коли зловмисник не може використати той самий канал для атаки та збору інформації, або коли сервер працює надто повільно чи нестабільно, щоб виконувати ці дії. Цей підхід

залежить від здатності сервера генерувати запити HTTP або DNS, щоб надіслати дані зловмиснику [6].

Вразливості SQL–ін'єкцій, атак і методів виникають різними способами. Приклади впровадження SQL включають:

– Екранування запиту:

Коли запит має складну структуру, до якої важко або навіть неможливо застосувати UNION, можна зіткнутися з цим типом атаки з використанням SQL–коду. Розглянемо наступний запит:

```
"SELECT Author FROM Journals WHERE Author_ID =
= id ANDAuthor_Name LIKE ( 'J%') "
```

У даному запиті Journals є деякою таблицею журналів, Author_ID – унікальний номер учасника журналу, Author_Name – ім'я автора журналу, а id – параметр, який передано. Тільки коли ім'я автора починається з букви «J», сервер створює вибірку імен авторів відповідно до переданого параметру ідентифікації. Екранування запиту використовується в цьому випадку, оскільки зловмиснику досить важко впровадити в запит UNION [7]. Тобто передається натупне значення параметра: –1 UNION SELECT usr_name, password FROM admin / *.

Після цього запит виглядатиме таким чином:

```
"SELECT Author FROM Journals WHERE Author_ID = –1 UNION SELECT
usr_name, password FROM admin / * AND Author_Name LIKE ( 'J%') "
```

Виконання цього запиту призведе до збору імен користувачів і паролів із таблиці адміністратора, оскільки автора з унікальним номером –1 швидше за все не існує, а частина запиту, яка обмежує вибірку по імені автора, була закоментована зловмисником. У результаті виконання частина запиту AND Author_Name LIKE ('J%') не впливає на виконання. Екранування запиту здійснюється різними символами коментування залежно від типу СКБД.

–Зміна вхідних параметрів шляхом додавання в них конструкцій мови SQL:

В даному випадку використовується SQL–код для запитів, які використовують параметр ідентифікації, тобто числовий тип, як вхідний параметр. Наприклад, взяти до уваги цю url–адресу:

```
http: //localhost/test_1? book_id = 1
```

Можемо зробити припущення, що буде згенеровано наступний запит до СКБД:

```
"SELECT * FROM Books WHERE Book_ID = id"
```

В цьому запиті використовуються таблиця книг, Book_ID — унікальний номер книги в таблиці книг, a id — параметр, отриманий сервером. Очевидно, що якщо на сервер надсилається параметр ідентифікації, який дорівнює , наприклад 80, то виконується наступний SQL–запит:

```
"SELECT * FROM Books WHERE Book_ID = 100"
```

Однак якщо передати на сервер в якості параметра id рядок "–1 OR 1 = 1", то вийде наступний SQL–запит:

```
"SELECT * FROM Books WHERE Book_ID = –1 OR 1 = 1"
```

В результаті виконання цього SQL–запиту сервер виведе всі наявні в базі книги, оскільки Book_ID = –1 є помилковою умовою, за замовчуванням, а умова 1 = 1 завжди є вірною. Підводячи підсумок, очевидно, що логіка всього SQL–запиту змінюється, коли змінюються вхідні параметри за допомогою SQL–ін'єкцій

- SQL–ін’єкції в рядкові параметри. Для цієї ін’єкції перше що можна розглянути це відмова від авторизації. Припустимо, код веб–додатка має наступний вигляд:

```
SQLQuery = "SELECT Username FROM Users WHERE Username = '"
&strUsername & " 'AND Password ='" & strPassword & " '"
strAuthCheck = GetQueryResult (SQLQuery)
If strAuthCheck = "" Then boolAuthenticated = False
```

```
Else
boolAuthenticated = TrueEnd If
```

Таким чином, можемо побачити що відбувається коли користувач вводить свій логін і пароль. Запит буде проходити через таблицю користувача, щоб перевірити, чи збігаються дані користувача з даними в таблиці. Якщо ці дані знайдені, змінна strAuthCheck зберігатиме ім'я користувача, що означає, що користувач повинен бути аутентифікований. Змінна strAuthCheck буде порожньою, якщо даних немає, і користувач не зможе провести перевірку справжньості [8].

Іншим випадком є впровадження коду в оператор LIKE. Припустимо, у нас є запит, виконання якого дозволяє користувачеві отримати вибірку статей. Якщо в назві статті є слово, необхідне для вибірки, стаття потрапляє до вибірки. Після цього вигляд SQL–запиту буде таким:

```
"SELECT Book_ID, Book _Date, Book _Caption, Book _Text,
Book _ID_Author FROM Books WHERE Book _Caption
LIKE ( '% Search_Text%') "
```

В даному запиті використовуються наступні параметри: Books — певна таблиця статей, Book_ID — унікальний номер книги, Book_Date — дата публікації книги, Book_Caption — опис книги, Book_Text — текст книги, Book_ID_Author — унікальний номер автора книги в таблиці книг, а Search_Text — параметр, який був переданий.

–Розщеплення SQL–запиту

Виконання кількох SQL–запитів замість одного запланованого запиту є метою такого типу атаки. Зловмисник використовує знак «;» (крапка з комою) для цього. Але ця можливість не підтримується в усіх версіях SQL.
Припустимо, ми отримуємо запит:

```
"SELECT * FROM Journals WHERE Journal_ID = id"
```

В даному запиті Journals – це окрема таблиця журналів, Journal_ID – це унікальний номер журналу в таблиці Journals, а id – це параметр, який сервер надає. Замість того, щоб передавати звичайне значення параметра, можна спробувати передати таке значення, наприклад:

```
"80; INSERT INTO admin (usr_name, password)VALUES ( 'AdmName',
'1234') "
```

Тоді запит матиме такий вигляд:

```
"SELECT * FROM Journals WHERE Journal_ID = 80; INSERT
INTOadminь(usr_name, password) VALUES ( 'AdmName', '4320') "
```

В цьому випадку в одному запиті будуть виконані 2 команди:"SELECT * FROM Journals WHERE Journal_ID = 80"

```
"INSERT INTO admin (usr_name, password) VALUES
( 'AdmName', '4320') "
```

У підсумку в таблицю admin буде додано запис про нового адміністратора.

## 1.2 Методи та засоби виявлення SQL–ін'єкцій

На сьогодні існує багато методів та засобів виявлення та запобігання SQL–ін'єкцій, більшість з яких відрізняються один від одного. Ці методи використовують спеціальні інструменти та механізми кодування, щоб виявити або запобігти зловмисним атакам SQL на цільову базу даних сервера.

Завдання запобігання атакам SQL–ін’єкції є дійсно складним питанням. Зловмисники завжди можуть отримати доступ до баз даних через лазівку в коді. Розпізнавання аномальної чи підозрілої діяльності називається виявленням мережевих атак. Дані для аналізу представляють мережевий трафік у формі мережевих пакетів. Ці дані можна нормувати для визначення ознакових атрибутів загального виду після їх збору без обробки. Створення активного профілю здійснюється за допомогою цих даних. Після цього профіль

порівнюється з нормальною діяльністю об’єкта, і якщо параметри профілю відрізняються, аномалія фіксується [9].

Багато способів використання цього алгоритму включають порівняння з граничною величиною, яке фіксує аномалію при перевищенні граничної величини, ідентифікація несанкціонованих дій шляхом порівняння мережевого трафіку з шаблоном атак або інтелектуальний аналіз даних, включаючи методи обчислювального інтелекту та машинного навчання. В таблиці 1.2 наведено типові методи боротьби з впровадженням SQL– ін’єкцій [9].

Таблиця 1.2 Методи боротьби з впровадженням SQL– ін’єкцій

| Метод | Переваги | Недоліки |
|---|---|---|
| Фільтрація цілочисельних параметрів | Якщо цілочисельні параметри використовуються в SQL-запитах, фільтрація може вберегти від потенційних SQL-ін'єкцій. | Якщо фільтрація реалізована неправильно або неповно, може виникнути можливість обхіду ін'єкцій або атак. |
| Скорочення вхідних параметрів | Може допомогти запобігти атакам SQL-ін'єкцій, оскільки потенційно шкідливі символи будуть видалені або обрізан | Скорочення може викликати втрату корисної інформації, оскільки частина введених даних може бути відсіяна |
| Використання параметризованих запитів | Введені дані не вбудовуються як частини SQL-запиту, а використовуються як параметри | При створенні динамічних SQL-запитів або використанні динамічних назв таблиць, може бути складно використовувати параметризовані запити |
| Використання принципу найменших привілеїв при наданні доступу до баз даних | Дозволяє обмежити доступ користувачів до необхідних ресурсів, що допомагає запобігти можливим вторгненням та зменшити ризик. | В умовах зі складною структурою баз даних може знадобитися більше часу для |

| | | |
|---|---|---|
| | | встановлення правильних прав доступу для кожного користувача |
| Регулярні вирази | Легко використовувати для пошуку певних патернів | Не завжди ефективні для всіх видів ін'єкцій, може виникати багато помилок. |
| ORM (Object-Relational Mapping) | Абстрагує взаємодію з базою даних, автоматично використовує параметризовані запити | Деякі ORM можуть мати свої вразливості |
| Валідація введених даних | Перевірка введених даних на допустимість може запобігти ін'єкціям | Не завжди ефективний і може бути обхідний |
| Хешування та шифрування паролів | Захищає від ін'єкцій, спрощує захист конфіденційних даних | Не вирішує всіх проблем ін'єкцій, може бути витікає |

Продовження таблиці 1.2

| | | |
|---|---|---|
| Firewalls за допомогою правил виявлення SQL-ін'єкцій | Блокує атаки на рівні мережі | Може дати хибні позитиви чи негативи, вимагає налаштувань |
| Системи виявлення вторгнень (IDS) та системи захисту від вторгнень (IPS) | Моніторинг та блокування ненормальної активності | Вимагає налаштувань, може викликати ложні тривоги |
| Використання спеціалізованих бібліотек та фреймворків | Забезпечує готові інструменти для захисту від ін'єкцій | Потребує інтеграції та оновлень |

Ефективність різних методів та засобів виявлення та захисту від SQL-ін'єкцій може залежати від конкретних обставин, технічних аспектів проекту та використовуваних технологій.

Розглянемо технологію SQL injection prevention system. Коли хакер намагається використати SQL–ін’єкцію на вразливому місці веб–сторінки, він

формує спеціальний запит на сервер. Тому необхідно перевіряти інформацію від користувачів, оскільки вхідні дані можуть бути небезпечними. Тому існує система профілактики, так звана перевірка перевірки. Це набір методів для фільтрації всіх вхідних даних які надходять від користувачів та генерують підпис запиту. Після того ін'єкції не можна використовувати, оскільки Secure Shell фільтрує та виводить інформацію а потім блокує впровадження в базу даних [10].

1. Екранування виводу. Оскільки серверна сторона Інтернету програми часто взаємодіє із веб–сторінками, URL–адресами та базами даних, існує багато функцій, які використовуються для обробки цих даних. Більшість інформації обробляється як тип рядка, але в різних ситуаціях (база даних або сервер) необхідно використовувати різні перевірки з фільтрами. Існують функції PHP для перетворення спеціальних символів у рядок у їхні сутності, видаливши теги HTML і вилучення мета–тегів [10].

2. Перевірка структури вхідних даних за допомогою регулярних виразів. Для цієї мети можна використовувати регулярний вираз, який є доступним способом визначення шаблону можливих рядків. Якщо розробник знає, що вхідні дані мають структуру, наприклад e–mail клієнта, телефон або дата – має сенс перевіряти значення за допомогою регулярних виразів. Таким чином, перевірено точність введених даних і одночасно відрізати всі небажані символи та неприйнятні модифікації [10].

3. Використання винятків. Щоб уникнути зупинки послуг, яка трапляється, коли виникає помилка, визначена веб–програмою необхідно змінити звичайний потік виконання коду. Така ситуація називається обробкою винятків. Іноді програмне забезпечення не виконуватиметься, як передбачено, що призведе до помилки. Є ряд причин, які можуть спричинити виключення, наприклад:

– на веб–сервері не вистачає місця на диску;

– користувач вводить неправильне значення в поле форми;

– файл або запис бази даних не існує;

– програмне забезпечення не має дозволу робити щось;

– послуга тимчасово/постійно недоступна [10].

Розглянемо використання технологій захисту від різних типів SQL–ін’єкцій:

– *Використання технологій захисту від Blind SQL Injection*

Виявлення та профілактика — це складні завдання. Однак легше запобігти нападам, якщо є правильне розуміння типів SQL–ін’єкції. Для того, щоб запобігти сучасним SQL–ін’єкціям, краще використовувати підготовлені шаблони, які є фіксованими та не можуть бути змінені користувачем веб–сайту або веб–додатку. Такі методи, як mafic_quotes() і add_slashes(), не можуть запобігти атакам на веб–додаток або веб–сайт. Отже, сдід розгляне кілька різних стратегій виявлення та запобігання сучасним SQL–ін’єкціям [11].

– Існує багато науково–дослідних робіт, які описують сліпі SQL–ін’єкції, які включають різні системи виявлення та методи запобігання впровадженню даного типу атак. Біла SQL–ін’єкція важко виявити та запобігти. Найпоширеніший метод AMNESIA, який використовується для аналізу, спостереження та нейтралізації SQL–ін’єкцій. Інструмент використовується лише для захисту Java–додатків і під час процесу моніторингу. Розробники радять використовувати алгоритми машинного навчання для покращення виявлення та запобігання сліпих SQL–ін’єкцій. Крім того, вони проаналізували результати та підтвердили, що запобігання та виявлення перевершили чистий SQLCheck і AMNESIA [11].

– *Використання технологій захисту від Fast Flux*

Основні атаки цього типу спрямовані на слабкі місця систем захисту клієнтів. Навіть системи захисту ФБР були стурбовані цим типом атак, оскільки з Fast Flux SQL–ін’єкції зіткнулися в Університеті штату Індіана, США. Збільшення безпеки серверів є найкращим способом захисту від будь–якої атаки. Метод, за допомогою якого точка URL доставки JavaScript потрапляє в чорний список, якщо вони визначені в стилі Fast Flux, дозволяє захистити

швидкий потік. Fast Flux Monitor (FFM) є прикладом практичної реалізації цього методу. Він може виявити та класифікувати Fast Flux Service Networks за лічені хвилини та використовувати як активний, так і пасивний моніторинг DNS, що доповнює довгострокове спостереження за FFSN [12].

Після класифікації Fast Flux мережі можна використовувати цей метод, щоб зупинити SQL–ін'єкцію за допомогою додаткового моніторингу. Для забезпечення методів подальшого кодування впровадження таких технологій розглядається як навідні заходи. Однак злочинці стають все розумнішими, і навіть дослідники контррозвідки ООН долучилися до розробки нових методів захисту. Вони зробили висновки щодо поля дослідження, виявлення швидких мереж потоку та SQL–ін'єкцій, а також запропонували технологію Expert Systems. Данна технологія використовує методи машинного навчання для пошуку SQL–ін'єкцій. Вони звертають увагу на класифікатор C5.0 і наївний байєсівський класифікатор для виявлення атаки SQL–ін'єкції Fast Flux [13].

– *Використання технологій захисту від SQLi XSS*

Ardilla Tool — інструмент для пошуку SQL+XSS атак. Він має два режими перевірки на наявність небезпеки: суворий і lenient. Ідентифікація та запобігання ін'єкціям використовуються під час жорсткого режиму [14].

Технологія роботи засобу використовує Taint Base, метод виявлення існуючих методів захисту системи. Використовуючи цей метод, Ardilla аналізує статичні методи, які вже використовувалися, і якщо попередні вимоги до застосування системи не виконані за допомогою наведеного вище методу, Ardilla додає необхідні методи в систему за допомогою фільтрів заповнення та так званої «санітарної» обробки. SQL–ін'єкції часто використовують разом із XSS і Java Script. Через те, що для повного захисту системи потрібно першочергово захистити XSS, перш ніж використовувати методи захисту SQL, як це робить Ardilla [15], подібні проекти не використовують такого ж рівня безпеки.

Котроль потоку виконання — це механізм, який використовується для

обробки обох етапів впровадження захисту. Його основна мета полягає в тому, щоб захистити SQL–ін'єкції, приховані в тілі XSS атаки. Для аналізу клієнтського коду Java Script використовується автомат кінцевого стану. Автомат обробки даних запобігає проникненню шкідливого коду в тілі скріпта в базу даних [16].

–Noxes Tool може бути ще одним прикладом реалізації методів захисту. З іншого боку, метод, описаний вище, не забезпечує такої ж безпеки, як інші, оскільки він вразливий до тегів html, які зловмисники можуть використовувати для впровадження шкідливого коду замість самих скріптів. Наявність функції контролю кукі, перевірка https запиту та запобігання зміні http заголовку є одними з переваг даного інструменту [17].

– *Використання технологій захисту від SQLi DNS*

Щоб захистити поєднання даних від типів атак, необхідно використовувати технологію розподілення. По–перше, ідентифікується втручання в DNS, а потім впроваджуються процедури захисту від ін'єкцій. У разі відмови у завантаженні, як приклад, безкоштовного програмного забезпечення з веб–сайтів, DNS Hijacking можна запобігти, оскільки дані продуктів мають багато вразливостей. Крім того, слід розглянути підміну DNS. Цей спосіб намагається взяти на себе налаштування роутера клієнта. Для впровадження запобіжних заходів був запропонований фільтр, який слідкує за потоком виконання DNS, який називається LWCSS (легкий захисний захист сторони клієнта) [18].

– *Використання технологій захисту від SQLi Cross Domain Policies*

Методи запобігання цьому типу атак включають правильне впровадження політик захисту та зменшення використання так званих субдоменних вразливостей. Наразі існує DEMACRO–метод. Він використовує як статичні, так і динамічні аналізатори коду, які використовуються в межах мережі. Перевагою цього методу є те, що він не вимагає використання машинного навчання для нормативного аналізу. Суть методу полягає в тому, щоб знайти спробу атаки та деавторизацію нападника [19].

– *Використання технологій захисту від SQLi DDoS*

Незважаючи на те, що темп атаки DDoS широко обговорювався, зловмисники все ще використовують деякі лазівки для атак на систему. Дослідники використовували таку технологію, як кластерний аналіз. Вона допомагає в ідентифікації атаки та легко визначає тип атаки на систему. Після фази виявлення починається фаза пом'якшення, де порівнюються різні типи атак. Слід зазначити, що атаки SQLI DDoS дуже поширені. Веб–сервери, програми та веб–сайти є предметами захисту [20].

– *Використання технологій захисту від SQLI Insufficient Authentication*

Адміністратор може використовувати метод криптографічних функцій Hash, який використовується для захисту від SQL–ін'єкцій і недостатньої автентифікації, щоб уникнути проблем з автентифікацією. У цьому методі додаються хеш–функції для поля ім'я користувача та пароля. Алгоритми хеш генерують хеш–функції автоматично. Тепер, коли користувач вводить свої дані користувача, хеш–функція створюється та надсилається на сервер для перевірки. Все, що відбувається тут, зберігається та передається за допомогою шифрування. Можливість втручання в базу даних дуже мала, якщо ім'я користувача та пароль однакові для всіх користувачів, які зберігаються в базі даних, порівняно з її хешем функцією [21].

Використання методів машинного навчання (МН) для захисту веб-додатків від SQL-ін'єкцій є ефективним інструментом виявлення SQL-ін'єкцій та має ряд переваг, такі як:

– Автоматична ідентифікація можливих атак та запобігання їх виконанню.

– Можливість враховувати ширший контекст та складні залежності.

– Поєднання різних методів для ефективного виявлення та блокування атак.

– Моделі можуть адаптуватися до нових видів атак і змінювати свою поведінку на основі навчання на нових даних.

– Моделі можуть враховувати контекст веб-додатка, щоб робити більш точні висновки про безпеку запитів.

Але існує і ряд недоліків:

– Машинне навчання вимагає великої кількості даних для навчання, що може бути важко надати, особливо в разі нових проектів.

– Можливість хибних позитивів (визначення атак там, де їх немає) та хибних негативів (не виявлення реальних атак).

– Системи машинного навчання потребують постійного оновлення для ефективного захисту.

– Якість моделі буде залежати від якості та кількості використаних для навчання даних.

– Інтеграція методів машинного навчання з іншими методами захисту може бути ключовою стратегією [22].

Для візуального та зручного порівняння ефективності різних методів виявлення та засобів протидії SQL–ін’єкціям розглянемо таблицю 1.3, яка створена для дослідження ефективності різних методів виявлення та засобів протидії SQL–ін’єкція.

Таблиця 1.3 – Аналіз технологій виявлення та запобігання SQL–ін’єкціям

| № | Методи та засоби виявлення та протидії SQL–ін’єкціям | Blind SQL Injection | FF_SQLI | SQLI_XSS | SQLI_DNS | SQLI_CDP | SQLI_DDOS |
|---|---|---|---|---|---|---|---|
| 1. | Криптографічне хешування функцій | p | x | x | x | x | x |
| 2. | Динамічний перезапис файлів cookie | x | x | + | p | x | x |
| 3. | Статичний аналіз коду | o | x | o | x | * | x |
| 4. | Моніторинг часу роботи | p | + | * | x | x | x |
| 5. | Ardilla Tool | x | x | o | x | o | x |
| 6. | Noxes | x | x | p | x | + | x |

Продовження таблиці 1.3

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 7. | Session Shield | x | x | * | p | x | x |
| 8. | AMNESIA | * | x | p | x | x | x |
| 9. | SQLMap | o | x | o | p | x | o |
| 10. | Fast Flux Monitor | x | o | x | o | x | x |

Символи, які використовуються у таблиці 1.2:

(*) – використовується як для виявлення, так і для попередження.

(o) – використовується тільки для виявлення.

(+) – використовується для відображення профілактичних заходів.

(x) – зображують, що методи або засоби не відповідають сучасним SQL–ін’єкціям (з точки зору запобігання та виявлення).

(p) – повинен бути застосований інший метод для досягнення повноти виявлення та профілактики.

## 1.3 Формалізація вимог та постановка задачі

За результатами аналізу існуючих методів захисту було встановлено, що використання методів машинного навчання в поєднанні з регулярними виразами є одним із найкращих засобів захисту веб–додатків від SQL–ін’єкцій, адже це дозволяє створити більш гнучку та адаптивну систему захисту, яка може ефективно протистояти сучасним видам атак. Методи машинного навчання використовують при створенні сучасних систем виявлення атак адже це перспективний напрямок розвитку в цій галузі, оскільки вирішення схожих завдань поступово підвищує точність рішень[23].

На основі аналізу даних, описаних раніше, було визначено наступні завдання:

– спроектувати архітектуру засобу щоб захистити веб–додаток від SQL–ін’єкцій;

– створити загальні алгоритми роботи для засобів захисту та окремі алгоритми роботи для кожного модуля;

– навчитися використовувати інструменти розробки та мову програмування;

– створити програмний засіб на основі архітектури та алгоритмів роботи, які були розроблені;

– виконати підключення до веб–сайту, що має в вразливості до атак типу SQL–ін'єкцій, та на його основі проводити тестування.

Програмне забезпечення повинно відповідати наступним вимогам:

- можливість виявляти SQL–ін’єкції у веб–застосунку;
- запис інформації про SQL–ін’єкцію до лог–файл;
- блокування запитів, що містять ін'єкцію та запис в лог–файл;
- повідомлення про SQL–ін’єкцію адміністратору веб–застосунку;
- запис інформації про SQL–ін’єкцію до бази даних;
- безперервна робота програмного засобу.

# 2 РОЗРОБКА ІНФОРМАЦІЙНОЇ ТЕХНОЛОГІЇ

## 2.1 Структура інформаційної технології

Інформаційна технологія – це комплекс методів і ресурсів, які використовуються для виконання конкретних завдань. Інформаційна технологія складається з процесів. Такий метод дозволяє швидко виявити проблеми на ранніх стадіях розробки [24].

Інформаційна технологія виявлення SQL–ін’єкцій складається з чотирьох процесів, кожен з яких виконує різні функції. Рисунок 2.1 демонструє схему процесів інформаційної технології.

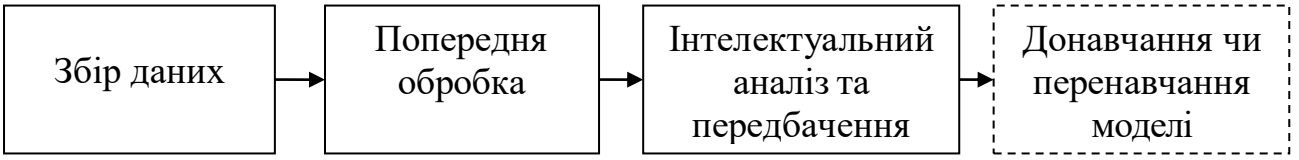



Рисунок 2.1 – Схема процесів інформаційної технології

Розглянемо окремо кожен процес.

1) Процес збору даних включає в себе такі етапи (рис. 2.2):

– отримання HTTP–запиту. На цьому етапі запит передається на веб–сервер;

– аналіз HTTP–запиту. На цьому етапі виконується аналіз структури HTTP–запиту. SQL–ін’єкція використовує вразливість безпеки, як правило, всередині або через веб–сервер, щоб використовувати базу даних SQL. Частини інструкцій SQL вводяться в поле веб–форми, а потім потрапляють на сервер бази даних, де вони обробляються.

- перевірка HTTP–запиту на основі шаблону. Для цієї мети використовуються регулярні вирази, які є доступним способом визначення шаблону можливих рядків.

Ось кілька регулярних виразів для виявлення SQL-ін'єкції в HTTP-запиті:

– Вираз для пошуку звичайних операторів SQL:

/\b(select|insert|update|delete|drop|truncate|create|alter)\b/

Цей вираз буде знаходити будь-який з цих операторів SQL у запиті[25].

– Вираз для пошуку символів, які часто використовуються для SQL-ін'єкції:

/\b(;|'|''|\||--|\.|*|%|\+|\-|\/|\|)\b/

Цей вираз буде знаходити будь-який з цих символів у запиті [25].

– Вираз для пошуку спеціальних символів, які часто використовуються для SQL-ін'єкції:

/\b(and|or|xor|not|in|between|like|is)\b/

Цей вираз буде знаходити будь-який з цих символів у запиті [25].

– Вираз для пошуку спеціальних символів, які часто використовуються для SQL-ін'єкції, у комбінації з оператором SQL:

/\b(select|insert|update|delete|drop|truncate|create|alter)\b\W(and|or|xor|not|in|between|like|is)\b/

Цей вираз буде знаходити комбінацію оператора SQL та одного з цих символів у запиті [25].

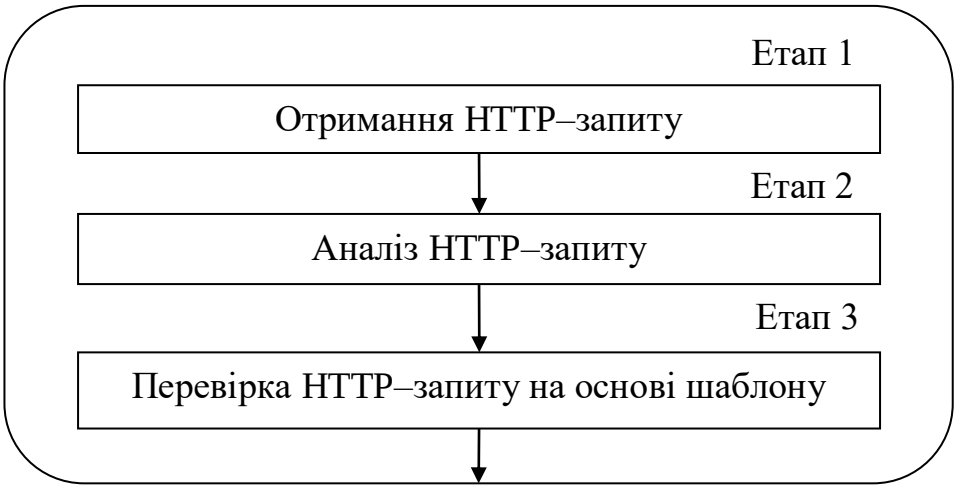



Рисунок 2.2 – Схема процесу збору даних

– Модель процесу можна описати як:

$$I_1 = <N_{http}, X_{check}>$$

де $I_1$ – результат перевірки HTTP–запиту за допомогою шаблонів, $N_{http}$ – отриманий HTTP–запит, $X_{check}$ – регулярний вираз.

2) Процес попередньої обробки включає в себе нормалізацію отриманих параметрів для підготовки цих даних до визначення моделлю машинного навчання.

Етап-моделювання неможливий без технічної підготовки даних. Така підготовка включає перевірку всіх типів даних і зведення їх до одного, щоб забезпечити найкращий доступ до аналізу обраних моделей і методів. Дані також потрібно збалансувати, що покращить результат тренування моделей машинного навчання.

Для роботи з даними їх необхідно перетворити у векторний формат. Векторне представлення дозволяє взаємодіяти з алгоритмами та застосовувати їх до різних типів завдань, таких як класифікація, регресія, кластеризація тощо. Перетворення даних у вектор є ключовим етапом у підготовці даних для застосування алгоритмів машинного навчання. Схему процесу наведено на рис. 2.3.

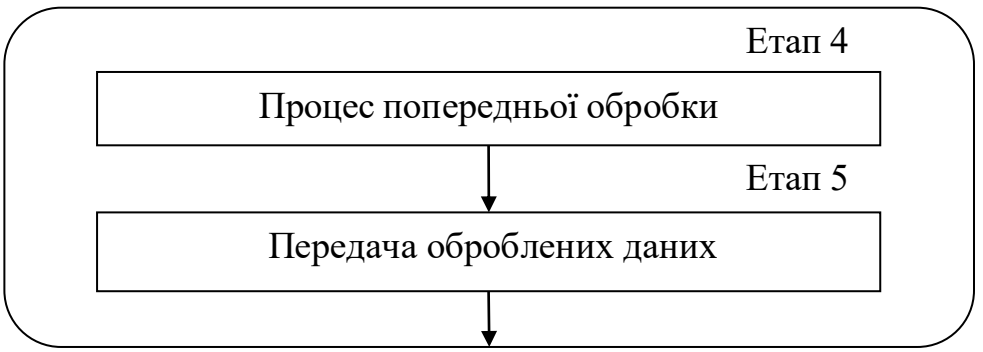



Рисунок 2.3 – Схема процесу попередньої обробки

Модель процесу можна описати як:

$$I_2 = \langle S_{sql}, V_{vectorize} \rangle$$

де $I_2$ – результат обробки даних, $S_{sql}$ – sql-запит, $V_{vectorizer}$ – векторайзер тексту.

Для виявлення SQL-ін'єкції в HTTP-запиті можна використовувати будь-який з регулярних виразів. Однак, оскільки ці вирази не є досконалими, вони можуть не знайти всі вразливі запити. Тому додатковим методом перевірки використовується модель машинного навчання.

3) Процес інтелектуального аналізу та передбачення. Під час інтелектуального аналізу моделі машинного навчання буде надано натреновану модель з відомими прикладами SQL–ін'єкцій, на підставі чого модель визначає ймовірність того, чи даний запит є SQL–ін'єкцією.


Рисунок 2.4 – Схема процесу інтелектуального аналізу та передбачення

Модель процесу можна описати як:

$$I_3 = < V_{vectorize}, K_{trained} >$$

де $I_3$ – результат класифікації, $V_{vectorizer}$ – векторайзер тексту, $K_{trained}$ – навчений класифікатор.

4) Донавчання чи перенавчання моделі

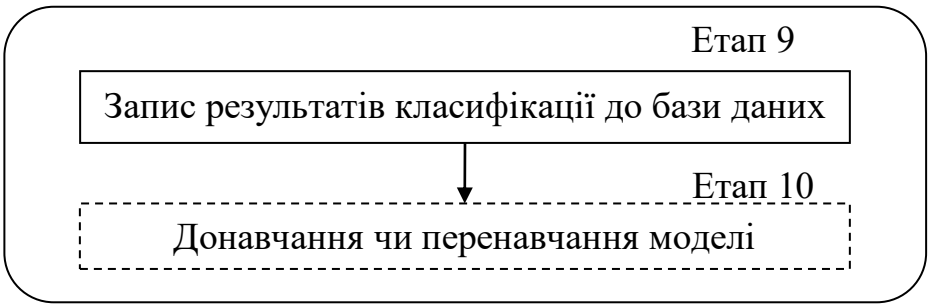



Рисунок 2.5 – Схема процесу донавчання чи перенавчання моделі

Модель процесу можна описати як:

$$I_4 = < D_{bd}, K_{trained} >$$

де $I_4$ – результат процесу донавчання чи перенавчання моделі, $D_{bd}$ – база даних, $K_{trained}$ – навчена модель.

Перенавчання моделі машинного навчання передбачає оновлення моделі новими даними або коригування її параметрів для покращення її продуктивності або адаптації до змін у базовому розподілі даних. Процес перенавчання має важливе значення для того, щоб модель залишалася актуальною та ефективною з часом. Ось загальні кроки для перенавчання моделі машинного навчання:

1. Збір нових, актуальних даних, які відображають поточні моделі та характеристики проблеми. Ці дані повинні охоплювати діапазон сценаріїв, з якими може зіткнутися модель.

2. Поєднання старих та нових даних з наявним набором даних, який використовується для навчання вихідної моделі. Цей об'єднаний набір даних буде використано для оновлення моделі.

3. Попередня обробка даних так само, як це було зроблено під час початкового навчання моделі. Це може включати очищення, обробку відсутніх значень, кодування категоріальних змінних і масштабування числових функцій.

4. Розділення даних на набори для навчання та тестування. Цей крок є вирішальним для оцінки ефективності моделі на невидимих даних.

5. Використання об'єднаних та попередньо оброблених даних, щоб перенавчити модель машинного навчання. Процес перенавчання може передбачати коригування існуючих параметрів моделі або використання нових даних для оновлення моделі.

6. Оцінка перенавченої моделі на наборі для тестування, щоб оцінити її продуктивність. Порівняння показники нової моделі з оригінальною моделлю та визначити, чи привело перенавчання до покращення результатів [26].

Частота повторного навчання залежить від характеристик проблеми, швидкості зміни основних даних і важливості точності моделі з часом. Регулярні оновлення та перенавчання мають вирішальне значення для того, щоб модель машинного навчання залишалася ефективною в динамічних середовищах [27].

.Модульна архітектура полегшує розробку, тестування, розуміння та підтримку програм. Кожен модуль може бути розроблений, протестований та оптимізований незалежно від інших, що сприяє швидкій ітерації та розвитку програмного продукту.

Програмне забезпечення для реалізації виявлення SQL–ін'єкцій складається з наступних модулів:

– модуль обробки HTTP–запитів;

– модуль інтелектуального аналізу;

– модуль збереження даних у базу даних.

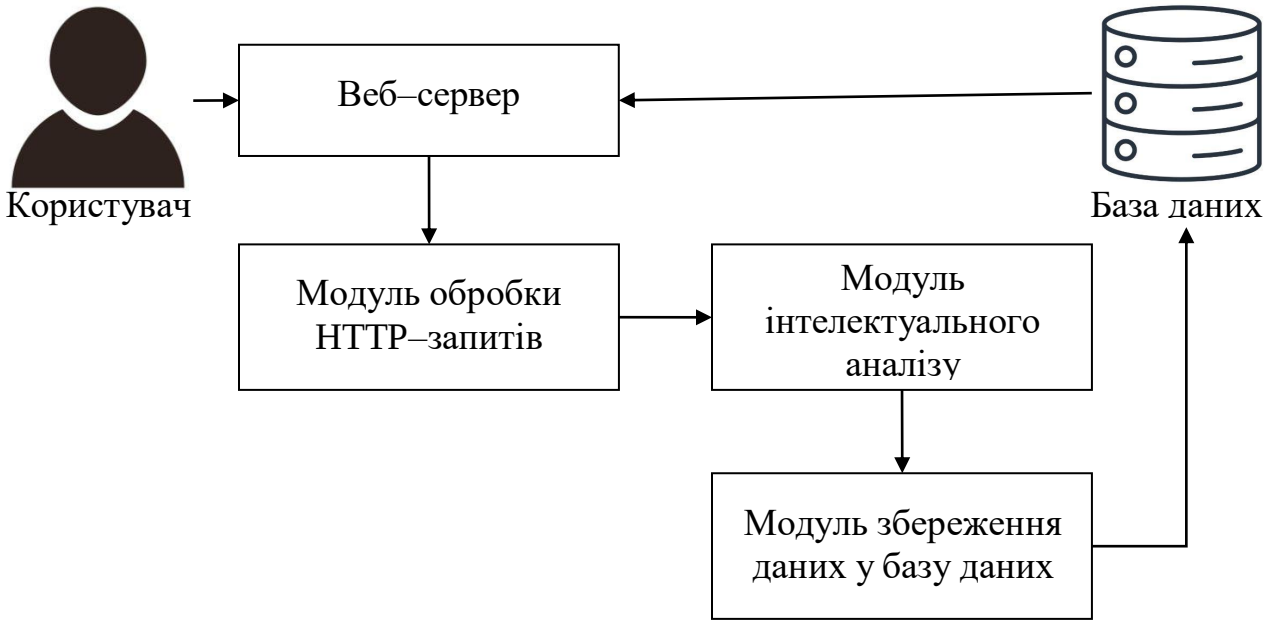



Рисунок 2.6 – Схема архітектури системи

Розглянемо кожен модуль системи окремо.

1) Створення модуля обробки HTTP–запитів передбачає відстеження та аналіз вхідних HTTP–запитів, щоб отримати уявлення про продуктивність веб–додатку, виявити потенційні проблеми та забезпечити безпеку системи. Для цього необхідно збирати основні показники, такі як:

– час запиту/відповіді;

– коди стану (успіх, помилка);

– методи запиту (GET, POST тощо);

– специфічні показники кінцевої точки;

– розміри запиту корисного навантаження;

– частота помилок;

– відстеження ненормальних шаблонів або високочастотних запитів.

Модуль моніторингу HTTP–запитів виконує аналіз структури HTTP–запиту та перевірку HTTP–запиту на основі шаблону. Наприклад, отримуємо запит:

1 or 8384 = like ( 'abcdefg',upper ( hex ( randomblob ( 500000000/2 ) ) ) )

Розберемо даний запит за структурою. У частині «1 or 8384 =» «1» і «8384» є умовами, які мають бути частиною пропозиції WHERE. Використання «or» означає, що впроваджена умова має бути істинною, можливо, щоб обійти перевірку автентифікації чи авторизації.

Частина «like ('abcdefg', upper(hex(randomblob(500000000/2))))» це спроба створити умову, яка включає ключове слово «LIKE». Він порівнює результат виразу «upper(hex(randomblob(500000000/2)))» з рядком «'abcdefg`». Частина «upper(hex(randomblob(500000000/2)))» генерує довгий шістнадцятковий рядок, а спроба ін’єкції намагається перевірити, чи він збігається з рядком «'abcdefg'». Метою цього, ймовірно, є створення дійсної умови, яка дозволяє виконувати введений код. Отже даний запит є SQL–ін'єкцією.

Розглянемо наступний приклад:

–1269" union all select 8412,8412,8412,8412,8412,8412,8412,8412—

Розберемо компоненти:

– «–1269"» це кінець числового значення або деяких інших даних у вихідному запиті.

– «union all select 8412,8412,8412,8412,8412,8412,8412,8412» ця частина намагається виконати операцію UNION для поєднання результатів вихідного запиту з указаними значеннями. У впровадженні UNION метою є відповідність кількості стовпців у вихідному запиті, а введені значення часто є заповнювачами для даних, які зловмисник хоче отримати.

– «—» Це коментар SQL, який вказує на те, що все після цього пункту розглядається як коментар і ігнорується механізмом бази даних.

Отже, коли цей рядок вставляється в уразливий SQL–запит, він має на меті додати новий оператор SELECT до вихідного запиту, об’єднуючи результати. Даний запит є прикладом спроби впровадження SQL–ін’єкції на основі UNION.

Отримємо наступний запит:

c/ virgen de la paloma, 107.

Проведемо аналіз. SQL–ін’єкція зазвичай передбачає додавання або маніпулювання кодом SQL у полях введення користувача таким чином, щоб вхідні дані інтерпретувалися базою даних як код SQL, а даний запит більше схожий на адресу чи частину даних. Отже, рядок «c/ virgen de la paloma, 107» не є SQL–ін’єкцією.

Проаналізуємо запит «SELECT AVG ( oldest ) FROM disappear SELECT SUM ( plates )»

Даний запит виглядає як два окремих оператори SQL, кожен з яких отримує дані зі своїх відповідних таблиць (зникнення та невизначена таблиця зі стовпцем під назвою plates). Наданий код не містить жодних ознак ін’єкції, це здається простим сценарієм SQL із двома операторами SELECT.

2) Модуль інтелектуального аналізу використовує навчену модель машинного навчання для виявлення SQL–ін'єкцій. Для вибору методу машинного навчання, який буде використано для виявлення SQL–ін'єкцій, необхідно провести моделювання різних методів машинного навчання, виконати аналіз результатів та відповідно до цього сформувати висновок.

3) Модуль збереження даних у базу даних виконує запис у файл для подальшого збільшення об’єму дата сету та покращення в майбутньому результатів передбачення моделі машинного навчання.

Програмний засіб, що реалізує інформаційну технологію виявлення SQL–ін'єкцій із застосуванням машинного навчання, виконує функції виявлення SQL–ін'єкцій та інформування користувача про наявність SQL–ін'єкцій.

Розглянемо алгоритм роботи програми (рис. 2.7).

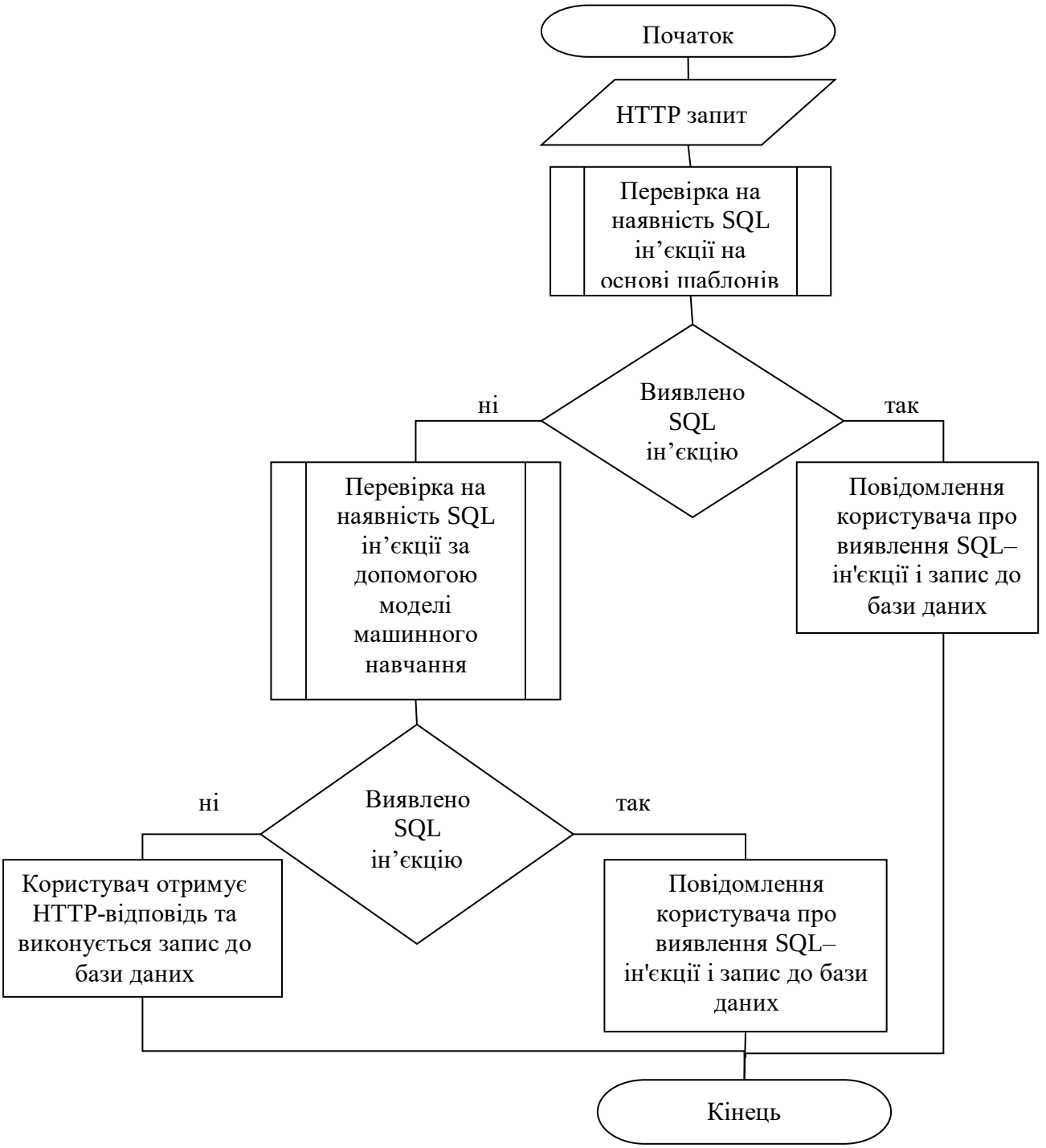



Рисунок 2.7 – Алгоритм роботи програми

1. Отримання параметрів HTTP–запитів – на цьому процесі виконується збір вхідних параметрів HTTP–запитів, які надсилаються до веб–застосункку.

2. Виконується перевірка на наявність SQL–ін’єкції на основі

шаблонів.

3. Якщо виявлено ознаки SQL–ін’єкції користувач отримує про це повідомлення, а запит записується до бази даних. Якщо ні то наступним кроком виконується перевірка за допомогою моделі машинного навчання.

4. За допомогою навченої моделі виконується знаходження зарезервованих слів мови SQL у параметрах HTTP–запитів, та якщо такі слова були знайдені, то даний запит буде вважатися SQL–ін’єкцією.

5. Формування даних про SQL–ін'єкцію – процес, на якому здійснюється формування інформації про SQL–ін’єкцію, для передачі цих даних до процесу сповіщення користувача про ін’єкцію та збереження до бази даних.

6. Якщо не виявлено ознак SQL–ін'єкції запит записується до бази даних і повертається назад до користувача.

### 2.2 Підготовка та збір даних для моделювання

Збір і підготовка даних для створення датасету – це важливий етап у багатьох проектах, таких як машинне навчання, аналіз даних, дослідження та багато інших.

Етап моделювання неможливий без технічної підготовки даних. Така підготовка включає в себе перевірку усіх типів даних та зведення їх до одного, що найкраще підходить для виконання аналізу обраними моделями та методами. Необхідно збалансувати дані, якщо вони цього потребують. Це покращить результат тренування моделей машинного навчання. І ключовий етап це розділення датасету на тренувальну та тестову вибірки. Такий крок дозволить оцінити якість побудованих моделей. Для навчання моделі машинного навчання було використано датасет SQL–ін'єкцій, який складається з 25000 записів (рис. 2.8) [28].

Перший стовпець містить необроблені рядки запиту SQL, а другий містить ціле значення 0 або 1, де 0 вказує на нешкідливий запит, а 1 вказує на шкідливий запит.

```
Перші 5 рядків:
                                                  Query  Label
0                    " or pg_sleep  (  __TIME__  )  --      1
1  create user name identified by pass123 tempora...      1
2   AND 1  =  utl_inaddr.get_host_address   (    ...      1
3   select * from users where id  =  '1' or @ @1 ...      1
4   select * from users where id  =  1 or 1#"  ( ...      1

Останні 5 рядків:
                                                  Query  Label
24995  DELETE FROM possible WHERE kitchen = 'largest'      0
24996                              DELETE FROM solution      0
24997             SELECT introduced ( s )  FROM agree      0
24998     SELECT * FROM  ( SELECT right FROM world )       0
24999                       SELECT TOP 3 * FROM century      0
```

Рисунок 2.8 – Структура датасету

Даний датасет є збалансованим, адже 11382 записів містять атаки, а 13618 не містять, що становить 46% та 54% (рис. 2.9).

```
print("Розмір датасету:", df.shape)
```

```
Розмір датасету: (25000, 2)
```

```
label_counts = df['Label'].value_counts()
print( "Безпечні SQL-запити:", label_counts [0], label_percentages[0], "%")
print( "SQL-ін'єкції:", label_counts [1], label_percentages[1], "%")
```

```
Безпечні SQL-запити: 13618 54.472 %
SQL-ін'єкції: 11382 45.528 %
```

Рисунок 2.9 – Результат перевірки збалансованості датасету

Дані, що передаються до класифікатора, зазвичай складаються з вхідних ознак і відповідних міток. У сценарії навчання під наглядом класифікатор навчається на позначеному наборі даних, де кожна точка даних пов’язана з певним результатом або міткою класу. Вхідні ознаки — це характеристики або атрибути даних, а мітки — це відповідні цільові значення або категорії, які класифікатор прагне передбачити.

Перетворення символьних даних у векторні дані є звичайним етапом попередньої обробки, адже більшість алгоритмів машинного навчання вимагають числових векторів як вхідних даних. Векторне представлення дозволяє легко взаємодіяти з цими алгоритмами та застосовувати їх до різних типів завдань, таких як класифікація, регресія, кластеризація тощо. Перетворення даних у вектор є ключовим етапом у підготовці даних для застосування алгоритмів машинного навчання та важливим елементом у багатьох завданнях аналізу даних.

Для порівняння було обрано такі класифікатори як CountVectorizer та TF-IDF

CountVectorizer і TfidfVectorizer є двома різними методами векторизації тексту, які використовуються у задачах обробки природної мови та машинного навчання. Обидва ці методи перетворюють текстові дані в числові вектори, але вони роблять це з різними підходами. Проведемо порівняння характеристик цих двох векторайзерів:

Таблиця 2.1 – Порівняння CountVectorizer і TfidfVectorizer

| | CountVectorizer | TfidfVectorizer |
|---|---|---|
| Опис | Перетворює текст у вектор, підраховуючи кількість входжень кожного слова в текстовому корпусі. | Використовує метод TF-IDF (Term Frequency-Inverse Document Frequency) для визначення ваги кожного слова у тексті. TF-IDF враховує як частоту слова в документі (TF), так і обернену частоту документа, в якому воно зустрічається (IDF). |
| Вага термін у | Використовується абсолютна кількість входжень слова в документ. | Інформаційна вага, яка враховує частоту слова та його рідкість в корпусі документів. |
| Призна -чення | Підходить для виявлення важливих слів в документі, але може надати великі значення словам, які часто зустрічаються в корпусі, але не є інформативними. | Дозволяє виділяти важливі слова, які зустрічаються часто в конкретному документі, але рідко в інших документах корпусу. Це може поліпшити якість векторизації для більш складних завдань. |

Обидва методи можуть бути використані для побудови матриць признаків, які використовуються у задачах машинного навчання, але вибір між CountVectorizer і TfidfVectorizer залежить від конкретної задачі та особливостей текстових даних.

Проведемо моделювання для класифікатора Decision Tree з використанням таких методів векторизації як CountVectorizer і TfidfVectorizer для порівняння результатів та визначення кращого векторайзера.

Матриці ознак грають ключову роль в задачах машинного навчання. Основна ідея полягає в тому, що вони представляють дані у вигляді числових значень, які можуть бути використані для тренування моделей машинного навчання (рис. 2.10).

```
print("CountVectorizer Matrix Shape:", X_train_count.shape)
print("TfidfVectorizer Matrix Shape:",  X_train_tfidf.shape)
```

```
CountVectorizer Matrix Shape: (20000, 20372)
TfidfVectorizer Matrix Shape: (20000, 20372)
```

Рисунок 2.10 – Розмір матриці для CountVectorizer і TfidfVectorizer

Векторизація даних (перетворення їх у вектори чи матриці) дозволяє використовувати оптимізовані бібліотеки для обчислень, що прискорює обчислення та поліпшує ефективність.

```
# Використовуємо CountVectorizer
count_vectorizer = CountVectorizer()
X_train_count = count_vectorizer.fit_transform(X_train)
X_test_count = count_vectorizer.transform(X_test)

# Класифікатор Decision Tree з CountVectorizer
dt_count_classifier = DecisionTreeClassifier(random_state=42)
dt_count_classifier.fit(X_train_count, y_train)
y_pred_count = dt_count_classifier.predict(X_test_count)
```

Рисунок 2.11 – Моделювання з використанням CountVectorizer

На рисунку 2.11 рображено фрагмент коду, в якому використовується

CountVectorizer для векторизації текстових даних. Потім виконується тренування класифікатора DecisionTreeClassifier на отриманих матрицях ознак. За отриманими даними можна провести оцінку результатів за допомогою метрик точності та classification report.

Виконаємо аналогічне моделювання для TfidfVectorizer (рис. 2.12).

```
# Використовуємо TfidfVectorizer
tfidf_vectorizer = TfidfVectorizer()
X_train_tfidf = tfidf_vectorizer.fit_transform(X_train)
X_test_tfidf = tfidf_vectorizer.transform(X_test)

# Класифікатор Decision Tree з TfidfVectorizer
dt_tfidf_classifier = DecisionTreeClassifier(random_state=42)
dt_tfidf_classifier.fit(X_train_tfidf, y_train)
y_pred_tfidf = dt_tfidf_classifier.predict(X_test_tfidf)
```

Рисунок 2.12 – Моделювання з використанням TfidfVectorizer

Для порівняння точності результатів моделювання класифікатора DecisionTreeClassifier розглянемо таблицю 2.2:

Таблиця 2.2 – Результати моделювання класифікатора DecisionTreeClassifier

| Назва векторайзера | Accuracy | Precision | Recall | F1 |
|---|---|---|---|---|
| CountVectorizer | 0,982 | 0,977 | 1,0 | 0,982 |
| TfidfVectorizer | 0,978 | 0,963 | 0,991 | 0,981 |

Проаналізувавши отримані результати можемо зробити висновок що в даному випадку доцільніше використовувати CountVectorizer

## 2.3 Моделювання та вибір класифікатора для вирішення задачі

Моделювання та вибір класифікатора — це важливі етапи в роботі над задачами машинного навчання. Для побудови моделей ми використовуємо машинне навчання з учителем. Ці методи автоматично будують модель

взаємозв’язків між набором описових ознак та цільовою ознакою на основі набору прецедентів – зафіксованих випадків. Після цього ми можемо використовувати цю модель для прогнозування.

Існують різні задачі, які можна вирішувати, використовуючи машинне навчання. В даному випадку потрібно вирішити задачу класифікації: задана певна множина об'єктів, для яких відомо, до яких класів вони належать. Ця множина називається навчальною вибіркою. Класова приналежність інших об'єктів не відома. Потрібно побудувати алгоритм, здатний класифікувати довільний об'єкт з початкової множини.

Визначення параметрів функції, прогнозування та тестування на одних і тих самих даних є методологічною помилкою: модель, яка б просто повторювала мітки зразків, які вона щойно бачила, мала б ідеальну оцінку, але не змогла б передбачити щось корисне на непомічених даних. Така ситуація називається перенавчанням. Щоб уникнути цього, загальноприйнятою практикою при проведенні експерименту машинного навчання з учителем є розділення на навчальну і тестову вибірки. Навчальний набір використовується для тренування моделі, тобто вивчання залежностей між ознаками та цільовою змінною. Тестовий набір використовується для оцінки ефективності моделі на нових даних, які не були використані під час тренування. Розділення на навчальний та тестовий набори допомагає забезпечити, що модель буде ефективною та загальним чином корисною на нових, раніше не бачених даних, що є головною метою будь-якої моделі машинного навчання. В даній роботі для навчання було обрано 80% даних, для тестування – 20% (рис. 2.13).

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

Рисунок 2.13– Розділення даних на тренувальну та тестову вибірки

Крос-валідація - це метод оцінки ефективності моделі машинного навчання, який дозволяє отримати більш надійні та стабільні результати порівняно з одноразовим розбиттям набору даних на навчальний та тестовий.

На рисунку 2.14 зображена схема типового крос–валідаційного процесу під час навчання моделей.

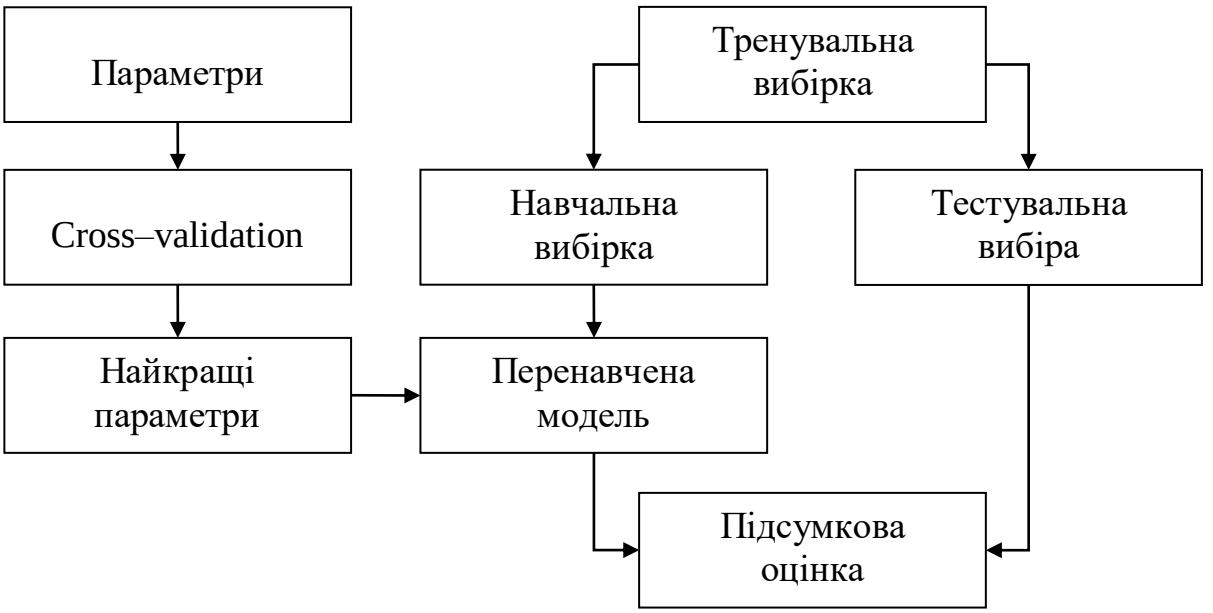



Рисунок 2.14 – Схема валідаційного процессу

Для вирішення задачі класифікації існує багато методів. Для аналізу було обрано наступні [23]:

–Support Vector Machine (SVM);

–K–Nearest Neighbors;

–Згорткова нейронна мережа (CNN);

– Gaussian Naive Bayes;

–DecisionTree;

–LogisticRegression.

Проведемо тестування для кожного з класифікаторів:

1) Gaussian Naive Bayes – вирішення задачі класифікації, базуючись на незалежності кожної пари ознак (дані для кожного представника обрані з простого розподілу Гаусса);

Першим кроком для моделювання необхідно визначити найкращий набір гіперпараметрів для кожного з класифікаторів. Одним із методів вирішення даної задачі є алгоритм GridSearch. Суть алгоритму – перебрати перелік гіперпараметрів моделі, виконати аналіз оцінки і отримати перелік параметрів,

при яких модель має найкращі результати (рис. 2.15).

```
param_grid = {'alpha': [0.1, 0.5, 1.0, 2.0]}

grid_search = GridSearchCV(naive_bayes_classifier, param_grid, cv=5, scoring='accuracy')
grid_search.fit(X_train_vectorized, y_train)

print("Best Hyperparameters for Naive Bayes:", grid_search.best_params_)
```

```
Best Hyperparameters for Naive Bayes: {'alpha': 0.1}
```

Рисунок 2.15 – Набір найкращих гіперпараметрів для класифікатора Naive Bayes

Далі виконуємо тренування моделі, використовуючи отриманий набір гіперпараметрів. Оцінкою ефективності класифікатора є значення accuracy, precision, recall, f1 (рис. 2.16).

```
accuracy = accuracy_score(y_test, y_pred)
print(f'Accuracy for Naive Bayes: {accuracy}')
```

```
Accuracy for Naive Bayes: 0.9712
```

```
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.97      0.97      0.97      2707
           1       0.97      0.97      0.97      2293

    accuracy                           0.97      5000
   macro avg       0.97      0.97      0.97      5000
weighted avg       0.97      0.97      0.97      5000
```

Рисунок 2.16 – Оцінка класифікатора Naive BayesНа Bayes

де accuracy – доля правильно визначених класів;

precision – відношення кількості правильно визначених класів до суми кількості правильно визначених класів та помилок першого роду (false positive);

recall – відношення кількості правильно визначених класів до суми кількості правильно визначених класів та помилок другого роду (false negative);

f1 – середньозважена оцінка похибок першого та другого роду;

Для дослідження ефективності класифікатора було побудовано матрицю помилок, яка є інструментом для оцінки відповідності між прогнозованими класами моделі та фактичними класами у задачах класифікації. Ця матриця відображає кількість правильних та неправильних класифікацій для кожного класу (рис. 2.17).

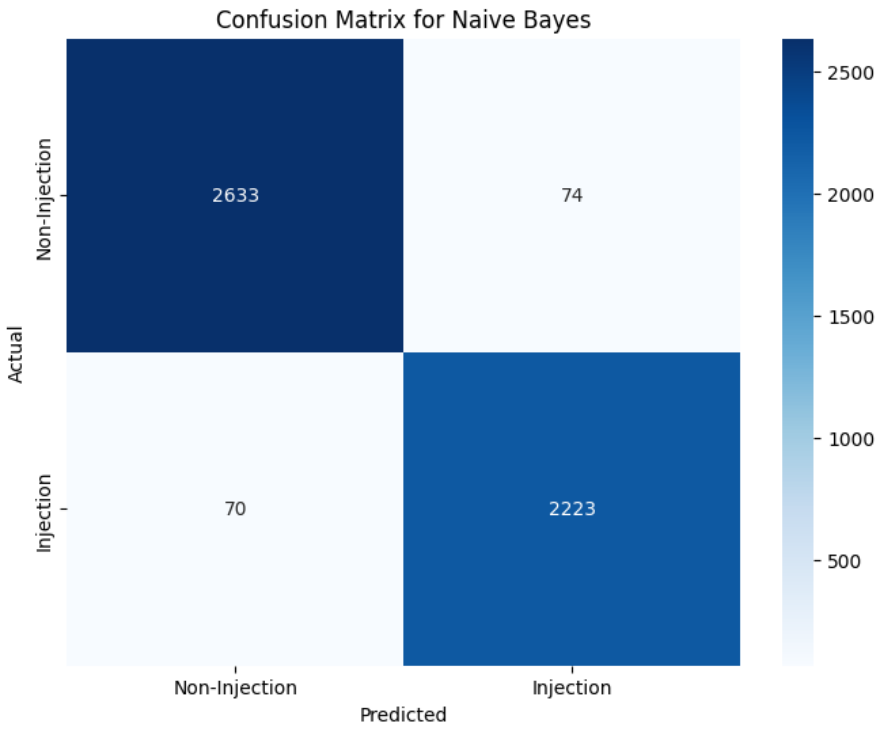



Рисунок 2.17 – Матриця помилок для класифікатора Naive Bayes

У контексті матриці помилок є чотири основні терміни:

– True Positive (TP) - кількість прикладів, які були правильно класифіковані як позитивні;

– True Negative (TN) - кількість прикладів, які були правильно класифіковані як негативні;

– False Positive (FP) - кількість прикладів, які були помилково класифіковані як позитивні (тип помилки I роду, або false alarm);

– False Negative (FN) - кількість прикладів, які були помилково класифіковані як негативні (тип помилки II роду).

Для визначення ефективності класифікатора було досліджено помилки першого і другого роду. Помилка першого роду полягає в тому, що буде

відхилена правильна гіпотеза. Помилка другого роду полягає в тому, що буде прийнята неправильна гіпотеза.

2) Support Vector Machine (SVM) – це контрольований алгоритм машинного навчання, який використовується як для класифікації, так і для регресії. Він належить до сімейства алгоритмів, які використовуються для задач навчання з учителем. SVM для класифікації (SVC) – використовується для вирішення задач класифікації, де дані розділяються на два або більше класів.

Першим кроком для оцінки класифікатора визначаємо найкращий набір гіперпараметрів (рис. 2.18).

```
param_grid = {'C': [0.1, 1, 10], 'kernel': ['linear', 'rbf'], 'gamma': ['scale', 'auto']}

grid_search = GridSearchCV(svm_classifier, param_grid, cv=5, scoring='accuracy')
grid_search.fit(X_train_vectorized, y_train)

print("Best Hyperparameters:", grid_search.best_params_)

Best Hyperparameters: {'C': 10, 'gamma': 'scale', 'kernel': 'linear'}
```

Рисунок 2.18 – Набір найкращих гіперпараметрів для класифікатора SVM

C (Constant) - параметр регуляризаці, менше значення C збільшує регуляризацію, що може призвести до більшої загальної гладкості границі рішень, але може також призвести до недооцінки слідів класів. Більше значення C дозволяє більше помилок на тренувальному наборі, але може покращити загальну ефективність на нових даних.

Виконуємо оцінку класифікатора (рис. 2.19):

```
accuracy = accuracy_score(y_test, y_pred)
print(f'Accuracy: {accuracy}')
```

```
Accuracy: 0.9808
```

```
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.98      0.99      0.98      2707
           1       0.98      0.97      0.98      2293

    accuracy                           0.98      5000
   macro avg       0.98      0.98      0.98      5000
weighted avg       0.98      0.98      0.98      5000
```

Рисунок 2.19 – Оцінка класифікатора SVM

SVM також включає елемент регуляризації, що допомагає запобігти перенавчанню та поліпшити загальну здатність моделі до нових даних.

Створюємо матрицю помилок (рис. 2.20):

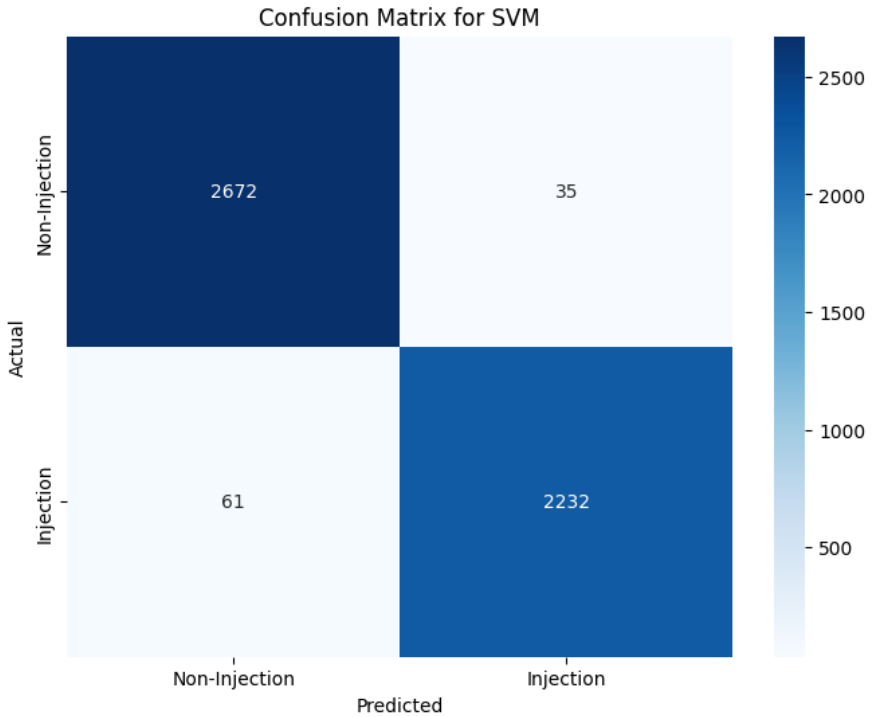



Рисунок 2.20. Матриця помилок для класифікатора SVM

3) K–Nearest Neighbors – встановлення мітки класу, використовуючи схожість ознак» найближчих k сусідів.

Визначаємо набір найкращих гіперпараметрів для класифікатора K–Nearest

Neighbors (рис. 2.21).

```
param_grid = {'n_neighbors': [3, 5, 7], 'weights': ['uniform', 'distance']}

grid_search = GridSearchCV(knn_classifier, param_grid, cv=5, scoring='accuracy')
grid_search.fit(X_train_vectorized, y_train)


print("Найкращі гіперпараметри:", grid_search.best_params_)
```

```
Найкращі гіперпараметри: {'n_neighbors': 3, 'weights': 'distance'}
```

Рисунок 2.21 – Набір найкращих гіперпараметрів для класифікатора KNeighbors

‘n_neighbors’– це основний параметр k-NN, який вказує, скільки найближчих сусідів враховувати під час класифікації. Вибір правильної кількості сусідів важливий, оскільки занадто мала кількість може призвести до недооцінки складності моделі, а занадто велика - до перенавчання.

Виконуємо оцінку класифікатора (рис. 2.22):

```
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)
```

```
Accuracy: 0.5902
```

```
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.96      0.25      0.40      2707
           1       0.53      0.99      0.69      2293

    accuracy                           0.59      5000
   macro avg       0.74      0.62      0.54      5000
weighted avg       0.76      0.59      0.53      5000
```

Рисунок 2.22– Оцінка класифікатора KNeighbors

Матриця помилок і відповідні метрики допомагають отримати повніше розуміння ефективності моделі в різних аспектах класифікації. Створюємо матрицю помилок (рис. 2.23):

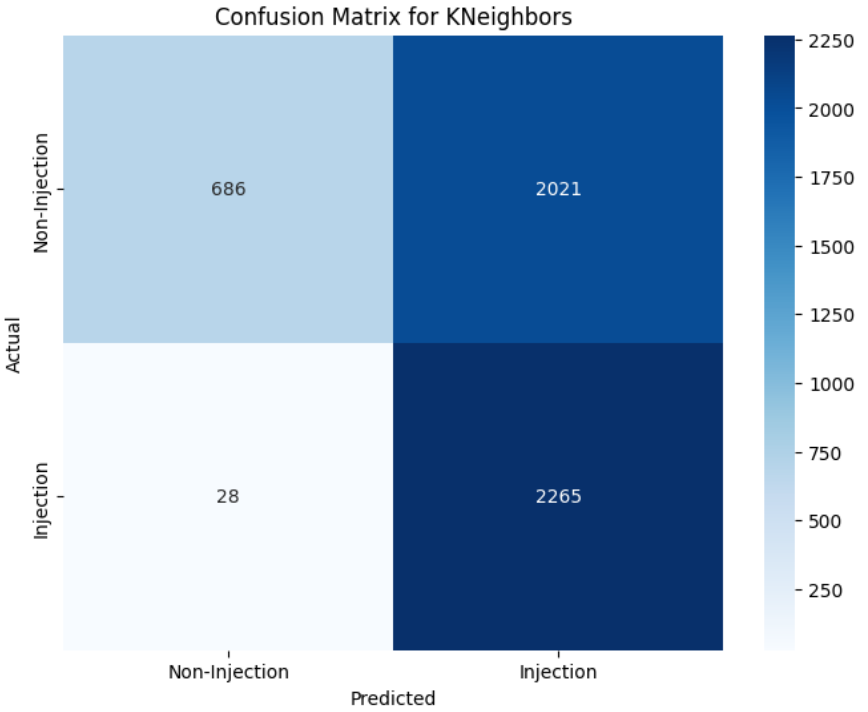



Рисунок 2.23 – Матриця помилок для класифікатора KNeighbors

Матриця помилок і відповідні метрики допомагають отримати повніше розуміння ефективності моделі в різних аспектах класифікації.

4) LogisticRegression – ймовірності описують можливі результати, використовуючи логістичну функцію. Логістична регресія у машинному навчанні використовується для бінарної класифікації, де модель прогнозує ймовірність того, що вхідний приклад належить до певного класу.

Визначаємо набір найкращих гіперпараметрів для класифікатора K–Nearest Neighbors (рис. 2.24).

```
param_grid = {'C': [0.001, 0.01, 0.1, 1, 10, 100], 'penalty': ['l1', 'l2']}

grid_search = GridSearchCV(logistic_regression_classifier, param_grid, cv=5, scoring='accuracy')
grid_search.fit(X_train_vectorized, y_train)

print("Найкращі гіперпараметри:", grid_search.best_params_)
```

```
Найкращі гіперпараметри: {'C': 100, 'penalty': 'l2'}
```

Рисунок 2.24 – Набір найкращих гіперпараметрів для класифікатора LogisticRegression

Виконуємо оцінку класифікатора (рис. 2.25):

```
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)
```

```
Accuracy: 0.9844
```

```
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.98      0.99      0.99      2707
           1       0.99      0.98      0.98      2293

    accuracy                           0.98      5000
   macro avg       0.98      0.98      0.98      5000
weighted avg       0.98      0.98      0.98      5000
```

Рисунок 2.25 – Оцінка класифікатора LogisticRegression

Оцінка класифікатора включає в себе аналіз його ефективності та визначення того, наскільки добре модель вирішує поставлені завдання класифікації.

Створюємо матрицю помилок (рис. 2.26):

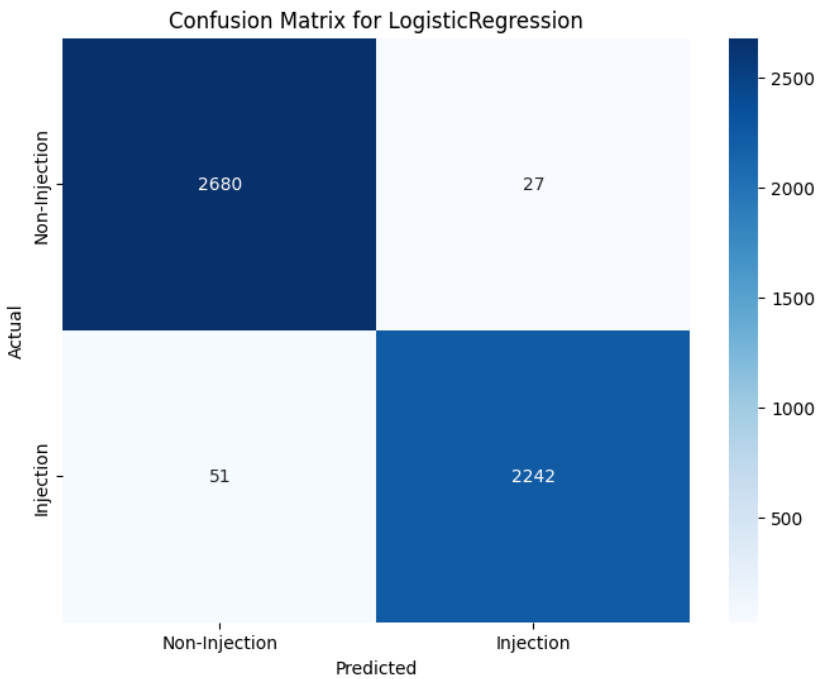



Рисунок 2.26–Матриця помилок для класифікатора LogisticRegression

5) Decision Tree (дерево рішень) - це модель машинного навчання, яка використовується для завдань класифікації та регресії. Вона працює, будуючи

дерево рішень або графік потоку прийняття рішень, на основі набору правил, що вивчаються з навчальних даних.

Визначаємо набір найкращих гіперпараметрів для класифікатора Decision Tree (рис. 2.27).

```
param_grid = {'criterion': ['gini', 'entropy'], 'max_depth': [None, 10, 20, 30]}

grid_search = GridSearchCV(decision_tree_classifier, param_grid, cv=5, scoring='accuracy')
grid_search.fit(X_train_vectorized, y_train)

print("Найкращі гіперпараметри:", grid_search.best_params_)

Найкращі гіперпараметри: {'criterion': 'entropy', 'max_depth': 30}
```

Рисунок 2.27 – Набір найкращих гіперпараметрів для класифікатора Decision Tree

Основні параметри для Decision Tree Classifier в бібліотеці Scikit-learn включають "criterion" – це функція для визначення якості розбиття, "entropy" - використовує ентропійний критерій та "max_depth" – це максимальна глибина дерева, визначає, наскільки глибоким може бути дерево рішень.

Виконуємо оцінку класифікатора (рис. 2.28):

```
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)

Accuracy: 0.9824

print(classification_report(y_test, y_pred))

              precision    recall  f1-score   support

           0       0.97      1.00      0.98      2707
           1       1.00      0.97      0.98      2293

    accuracy                           0.98      5000
   macro avg       0.98      0.98      0.98      5000
weighted avg       0.98      0.98      0.98      5000
```

Рисунок 2.28– Оцінка класифікатора Decision Tree

Створюємо матрицю помилок для класифікатора Decision Tree (рис. 2.29):

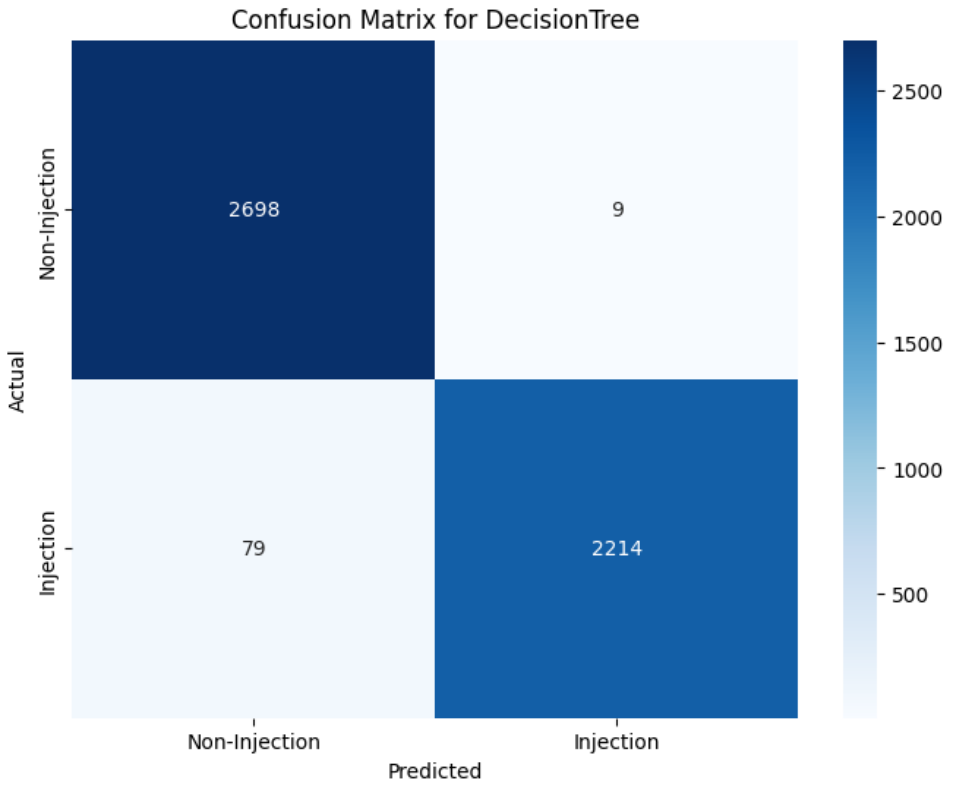



Рисунок 2.29 – Матриця помилок для класифікатора Decision Tree

6) CNN (Convolutional Neural Networks) - це тип глибоких нейронних мереж, які спеціалізуються на обробці та аналізі структурованих даних, таких як зображення. Вони зазвичай використовуються для завдань класифікації, визначення об'єктів, виокремлення ознак та інших завдань у сфері комп'ютерного зору.

Визначаємо набір найкращих гіперпараметрів для класифікатора CNN (рис. 2.30).

```
model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])

epochs = 10
batch_size = 32
```

Рисунок 2.30 – Набір найкращих гіперпараметрів для класифікатора CNN

Виконуємо оцінку класифікатора (рис. 2.31):

```
accuracy = accuracy_score(y_test, y_pred)
print("\nAccuracy:", accuracy)
```

```
Accuracy: 0.9924
```

```
print("\nClassification Report:")
print(classification_report(y_test, y_pred))
```

```
Classification Report:
              precision    recall  f1-score   support

           0       0.99      1.00      0.99      2707
           1       1.00      0.98      0.99      2293

    accuracy                           0.99      5000
   macro avg       0.99      0.99      0.99      5000
weighted avg       0.99      0.99      0.99      5000
```

Рисунок 2.31 – Оцінка класифікатора CNN

Створюємо матрицю помилок для класифікатора CNN (рис. 2.32):

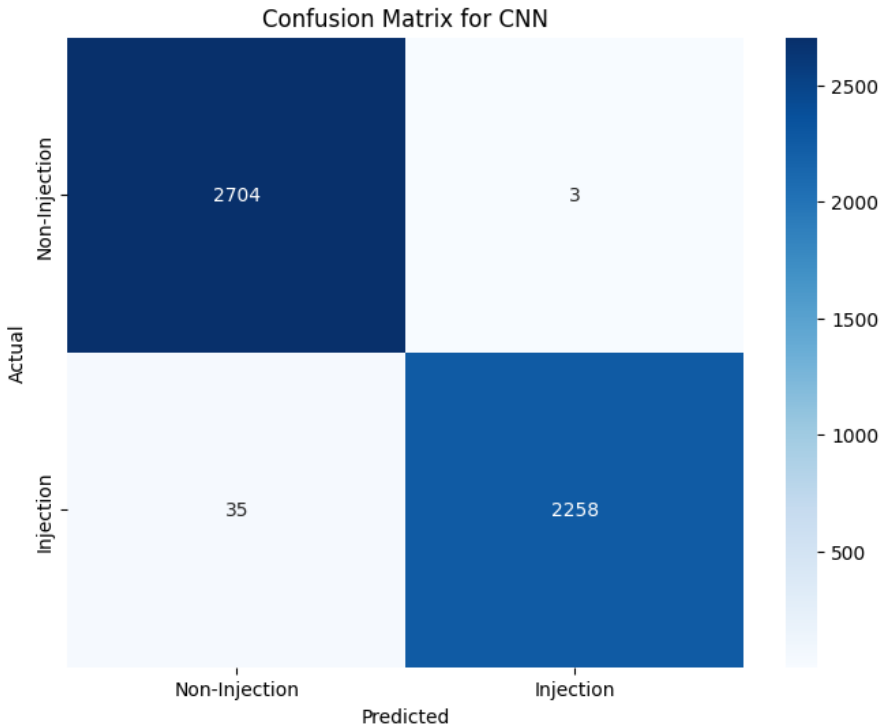



Рисунок 2.32– Матриця помилок для класифікатора CNN

Для дослідження було виведено помилки першого і другого роду у відсотках, адже це дієвий спосіб оцінити ефективність системи кібербезпеки і здійснити аналіз її функцій. Такий підхід дозволяє отримати кількісну оцінку того, наскільки часто система робить помилки того чи іншого типу в порівнянні з усіма виявленими подіями.

Для оцінки ефективності було використано такі показники як:

– False Positive Rate (FPR): Відношення кількості помилково позитивних визначень (FP) до загальної кількості нормальних подій. Це дозволяє оцінити, наскільки часто система помилково визначає нормальні події як загрози.

– False Negative Rate (FNR): Відношення кількості помилково негативних визначень (FN) до загальної кількості реальних загроз. Це показує, наскільки часто система пропускає реальні загрози.

Виведення помилок у відсотках дозволяє ідентифікувати слабкі місця системи та визначити напрямки для покращення.

Для вибору найкращого класифікатора необхідно порівняти результати класифікації усіх досліджених класифікаторів. Результати моделювання наведено у табл. 2.3.

Таблиця 2.3 – Результати моделювання класифікаторів

| Назва класифікатора | Accuracy | F1 | FP | FN | FPR | FNR |
|---|---|---|---|---|---|---|
| GaussianNB | 0,973 | 0,975 | 70 | 74 | 3,1% | 2,7% |
| SVM | 0,981 | 0,988 | 61 | 35 | 2,7% | 1,3% |
| KNearestNeighbors | 0,599 | 0,691 | 28 | 2021 | 1,2% | 76,7% |
| Logistic Regresion | 0,986 | 0,982 | 51 | 27 | 2,2% | 0,99% |
| DecisionTree | 0,983 | 0,987 | 79 | 9 | 3,4% | 0,33% |
| CNN | 0,991 | 0,991 | 35 | 3 | 1,5% | 0,1% |

Як бачимо, з таблиці 2.3 класифікатор CNN має найвищу точність визначення та швидкість навчання моделей., тому його було обрано для виявлення SQL-ін'єкцій.

# 3 ПРОГРАМНА РЕАЛІЗАЦІЯ ІНФОРМАЦІЙНОЇ ТЕХНОЛОГІЇ

## 3.1 Обґрунтування вибору інструментальних засобів розробки

Вибір інструментальних засобів (бібліотек мови програмування, модулів, програмних засобів) для реалізації основних модулів і спроектованої системи в цілому є першим етапом програмної реалізації інформаційних технологій.

Для зозробки засобу було обрано використання мови програмування високого рівня Python. Серед розробників програм машинного навчання вона є найпоширенішою мовою програмування. Вона містить набір інструментів, які були попередньо налаштовані, щоб дозволити впроваджувати моделі та алгоритми машинного навчання. Наявність великої кількості бібліотек і простий синтаксис є перевагами цієї мови. Розглянемо переваги і недоліки мови Python.

Переваги:

– Синтаксис Python зрозумілий і легкий для читання, що підкреслює читабельність і знижує вартість обслуговування програми.

– Python має величезну стандартну бібліотеку та багату екосистему бібліотек сторонніх розробників, що робить його придатним для широкого спектру програм.

– Python є універсальною мовою, і її можна використовувати для різних програм, включаючи веб–розробку, науку про дані, машинне навчання, штучний інтелект, автоматизацію тощо.

– Python сумісний з основними операційними системами, дозволяючи розробникам писати код, який може працювати в Windows, macOS і Linux без змін.

– Python є безкоштовним і відкритим кодом, що сприяє співпраці та широкому застосуванню.

– Численні онлайн–ресурси, форуми та спільноти, де розробники Python

можуть звернутися за допомогою та поділитися знаннями.

Недоліки:

– Python є інтерпретованою мовою, і, хоча він перевершує швидкість розробки, він може бути не таким швидким, як C або C++. Критичні для продуктивності програми можуть потребувати оптимізації або інтеграції з мовами низького рівня.

–. Python може бути не найкращим вибором для завдань, що потребують інтенсивного використання пам’яті, або програм, де низька рівень взаємодії системи є вирішальним.

– Пакування та розповсюдження програм Python може бути складним завданням, особливо коли мова йде про залежності та різні середовища.

– Python використовує динамічну типізацію та інтерпретується, що може призвести до помилок під час виконання, які могли бути виявлені під час компіляції в мовах зі статичною типізацією.

–Не завжди підходить для високопродуктивних обчислень (HPC). Для додатків, які вимагають надзвичайно високої обчислювальної продуктивності, такі мови, як Fortran або C, можуть бути більш придатними.

Таким чином, Python є потужною та універсальною мовою, яка широко використовується для різноманітних програм. Хоча він має деякі обмеження, його сильні сторони часто переважують слабкі, що робить його популярним вибором серед розробників для різноманітних проектів.

Виявлення SQL–ін’єкції за допомогою машинного навчання в Python передбачає використання бібліотек і методів машинного навчання для навчання моделі, яка може розрізняти нормальні та зловмисні запити SQL.

Для роботи було використано бібліотеки Pandas і Numpy. Numpy — це бібліотека для числових операцій у Python. Його часто використовують разом із pandas для ефективної обробки даних.

Pandas – це бібліотека Python для обробки та аналізу структурованих даних, її назва походить від «panel data» («панель даних»). Панельними

називаються дані, отримані в результаті дослідження і структуровані у вигляді інформації таблиці. Для роботи з такими масивами даних і створені Pandas. Дана бібліотека має такі властивості:

–додає структури даних та інструменти, призначені для роботи з табличними даними;

–надає інструменти для маніпулювання даними: зміни форми, об’єднання, сортування, нарізки, агрегація тощо;

–дозволяє працювати з відсутніми даними.

Також використовувалась бібліотека scikit–learn – це популярна бібліотека машинного навчання, яка надає інструменти для попередньої обробки даних, вибору функцій, навчання моделі та оцінювання. Він включає різні алгоритми для класифікації, що робить його придатним для створення моделі для виявлення SQL–ін’єкцій.

Для розробри інформаційної технології було обрано середовище Google Colaboratory, також відомий як Google Colab – це хмарна платформа, надана Google, яка дозволяє користувачам писати та виконувати код Python у спільному та інтерактивному середовищі. Ось кілька причин, чому Google Colab вважається хорошим і широко використовуваним:

– Однією з найважливіших переваг Google Colab є безкоштовний доступ до графічних процесорів (GPU) і тензорних процесорів (TPU). Це особливо корисно для завдань машинного навчання, які вимагають значної обчислювальної потужності, дозволяючи користувачам тренувати моделі швидше, не потребуючи дорогого апаратного забезпечення.

– Будучи хмарною платформою, Colab дозволяє користувачам отримувати доступ до своїх блокнотів і працювати з ними з будь–якого пристрою з підключенням до Інтернету. Крім того, він підтримує співпрацю в режимі реального часу, що полегшує роботу кількох користувачів на одному ноутбуку одночасно.

–У Colab попередньо встановлено багато популярних бібліотек і

фреймворків Python, які використовуються в науці про дані та машинному навчанні, наприклад NumPy, Pandas, TensorFlow і PyTorch. Це позбавляє користувачів від необхідності встановлювати ці бібліотеки вручну.

–Colab тісно інтегровано з Google Drive, що дозволяє користувачам зберігати свої блокноти безпосередньо на Google Drive. Ця інтеграція спрощує процес зберігання, спільного використання та організації коду та даних.

–Блокнотами Colab можна легко ділитися з іншими, а користувачі можуть публікувати свої блокноти в Інтернеті. Це робить його зручним для обміну кодом, результатами аналізу та досліджень із ширшою аудиторією.

–Colab спрощує доступ до даних і імпорт із різних джерел, зокрема Google Диска, Google Таблиць і зовнішніх URL–адрес. Це полегшує бездоганну інтеграцію даних у ваші проекти аналізу чи машинного навчання.

Хоча Google Colab пропонує численні переваги, важливо пам’ятати про його обмеження, такі як тимчасовий характер сеансів і обмеження на використання ресурсів. Для тривалих або ресурсомістких завдань користувачам може знадобитися розглянути альтернативні платформи хмарних обчислень або спеціальне обладнання.

Для тестування програмного засобу було розроблено веб–сайт з використанням HTML і CSS для інтерфейсу та Python для серверної частини. HTML (HyperText Markup Language) використовується для структурування вмісту веб–сторінки, визначення таких елементів, як заголовки, абзаци, списки тощо. CSS (каскадні таблиці стилів) використовується для стилізації та представлення. Це дозволяє розробникам контролювати макет, зовнішній вигляд і швидкість реагування веб–сторінок. Даний підхід є поширеним і ефективним у веб–розробці. Такий поділ проблем, відомий як архітектура переднього і заднього планів, вона використовує сильні сторони кожної технології у відповідних областях. Ось чому ця комбінація популярна.

Python зазвичай використовується на сервері для обробки серверної логіки. Це включає обробку даних, взаємодію з базами даних, обробку

автентифікації користувачів і керування загальним потоком програми. Правильно налаштований і захищений Python може допомогти у створенні безпечних серверних систем. Це дозволяє розробникам застосовувати найкращі методи автентифікації користувачів, перевірки даних і захисту від поширених веб–уразливостей.

Python може підключатися до різних баз даних, що робить його придатним для керування зберіганням і пошуком даних. Поширені системи баз даних включають MySQL, PostgreSQL, SQLite та MongoDB.

В даній роботі було обрано використання MySQL як бази даних для веб–сайту MySQL — це система керування реляційною базою даних (RDBMS) із відкритим кодом, яка широко використовується для веб–розробки завдяки своїй надійності, продуктивності та простоті використання. Ось кілька ключових моментів, які слід враховувати при використанні MySQL для веб–сайту:

–MySQL відома своєю надійністю та стабільністю. Він використовується у виробничих середовищах протягом багатьох років, і йому довіряють велика кількість розробників та організацій.

–MySQL оптимізовано для високопродуктивних операцій читання та запису. З належним індексуванням і оптимізацією запитів він може ефективно обробляти великі набори даних.

–MySQL порівняно легко налаштувати та керувати нею. Він має зручний інтерфейс командного рядка, а також графічні інтерфейси користувача (GUI), такі як MySQL Workbench для адміністрування та розробки.

–MySQL є відкритим вихідним кодом і вільним для використання, що робить його економічно ефективним вибором для малих і великих проектів.

–MySQL широко підтримується різними мовами програмування, веб–фреймворками та системами керування контентом (CMS). Це полегшує інтеграцію з різними компонентами вашого сайту.

Таким чином, MySQL є надійним і широко поширеним вибором для баз даних веб–розробки. Дотримуючись найкращих практик щодо дизайну баз

даних, безпеки та оптимізації, ви можете створити надійну та ефективну веб–програму, використовуючи MySQL як базову базу даних.

## 3.2 Програмна реалізація

Для реалізації інформаційної технології виявлення SQL–ін'єкцій розроблено модель машинного навчання, за допомогою якої виконується перевірка.

Першим кроком необхідно завантажити необхідні для бібліотеки (рис. 3.1).

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
from sklearn.preprocessing import LabelEncoder
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Embedding, Conv1D, GlobalMaxPooling1D, Dense, Dropout
from tensorflow.keras.callbacks import EarlyStopping
```

Рисунок 3.1 – Завантаження бібліотек

На рисунку 3.2 наведений процес створення моделі CNN за допомогою бібліотеки Keras.

```
model = Sequential()
model.add(Embedding(max_words, 50, input_length=max_len))
model.add(Conv1D(128, 5, activation='relu'))
model.add(GlobalMaxPooling1D())
model.add(Dense(64, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(1, activation='sigmoid'))
```

Рисунок 3.2 – Створення моделі CNN

Навчання моделі машинного навчання - це процес, під час якого модель "вчиться" на основі навчальних даних з метою виконання конкретного завдання. На рисунку 3.3 зображений процес навчання моделі машинного навчання:

```
history = model.fit(X_train_pad, y_train, epochs=epochs, batch_size=batch_size, validation_split=0.2, callba

Epoch 1/10
500/500 [==============================] - 16s 29ms/step - loss: 0.1015 - accuracy: 0.9697 - val_loss: 0.0243
Epoch 2/10
500/500 [==============================] - 14s 29ms/step - loss: 0.0215 - accuracy: 0.9961 - val_loss: 0.0223
Epoch 3/10
500/500 [==============================] - 15s 31ms/step - loss: 0.0165 - accuracy: 0.9969 - val_loss: 0.0256
Epoch 4/10
500/500 [==============================] - 14s 28ms/step - loss: 0.0135 - accuracy: 0.9978 - val_loss: 0.0262
```

Рисунок 3.3 – Навчання моделі

. Навчена модель зберігається, і викликається коли на веб–сервер надходить запит. Відповідно до результатів класифікації, якщо відповідь = 0 то виконується запит, якщо ні, то користувач отримує повідомлення: «Signs of SQL injection detected» (рис. 3.4).

```
load = open('/content/drive/MyDrive/SQL_injctions/my_model.pkl', 'rb')
load_model = pickle.load(load)

count_vect = CountVectorizer()
df = pd.read_csv('/content/drive/MyDrive/SQL_injctions/sql.csv')
count_vect.fit(df.text)

check = pd.DataFrame({'text': [message.text]})
check2 = count_vect.transform(check.text)
load_model.predict(check2)


if load_model.predict(check2) == 0:
    return "Login successful!"
else:
    return "Signs of SQL injection detected"
```

Рисунок 3.4– Функція перевірки запиту за допомогою моделі машинного навчання

Веб-сайт розроблений для перевірки та тестування має клієнтську та серверну частини. Клієнтська частина включає веб-сторінки, які користувачі відкривають у своєму браузері. HTML визначає структуру сторінок, CSS відповідає за їхній стиль та вигляд. Серверна частина обробляє запити від браузера та надсилає відповіді. Веб-сервер отримує запити, наприклад, відкриття сторінки чи надсилання даних форми, та повертає відповіді, які можуть бути HTML-сторінками, зображеннями, тощо. На рисунку 3.5 наведено структуру проекту.

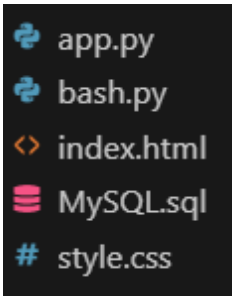



Рисунок 3.5 – Структура веб–додатку

HTML – визначає структуру веб-сторінки. Розглянемо фрагмент файлу index.html (рис. 3.6):

```
<body>
    <img src="201909_Security_SQL-Injection_1.png" width="420" height="100"
    <div class="container">
        <form action="app.py" method="post">
            <label for="username">Username:</label>
            <input type="text" id="username" name="username" required>
            <br>
            <label for="password">Password:</label>
            <input type="password" id="password" name="password" required>
            <br>
            <button type="submit">Login</button>
        </form>
    </div>
```

Рисунок 3.6 – Фрагмент класу body

CSS – відповідає за стилізацію та вигляд сторінки. Розглянемо фрагмент файлу style.css (рис. 3.7):

```
body {
    font-family: Arial, sans-serif;
    background-color: #f4f4f4;
    margin: 0;
    display: flex;
    justify-content: center;
    align-items: center;
    height: 100vh;
}
```

Рисунок 3.7 – Фрагмент стилізації класу body

Для запису запитів які надходять до веб-серверу необхідно створити базу даних (рис. 3.8):

```
CREATE DATABASE sql_injection;
USE sql_injection;

CREATE TABLE users (
    id INT AUTO_INCREMENT PRIMARY KEY,
    username VARCHAR(255) NOT NULL,
    password VARCHAR(255) NOT NULL
);
```

Рисунок 3.8 – Фрагмент коду створення бази даних sql_injection

На рисунку 3.9 зображена структура бази даних sql_injection:

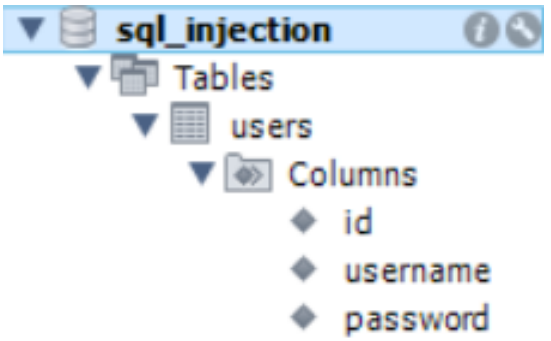



Рисунок 3.9 – Структура бази даних

Дані записуються у вигляді таблиці, фрагмент якої зображено на рисунку 3.10.

| id | username | password |
|---|---|---|
| 1 | daria | 1111 |
| 2 | user | 1235lom |
| 3 | juliana | 555555 |
| 4 | alex | alex123 |
| 5 | stepan | vin4200 |

Рисунок 3.10 – Фрагмент таблиці з даними користувачівув

Текст програмного засобу наведено у додатку Б. Для перевірки розробленого програмного засобу необхідно провести тестування.

### 3.3 Тестування програмного засобу

Тестування програмного засобу виконується з метою перевірки якості, правильності роботи і відповідності вимогам розробленого додатку. Тестування

дозволяє переконатися, що програмний засіб відповідає встановленим вимогам та функціональним характеристикам, дозволяє визначити, як швидко та ефективно програма виконує свої завдання. Це важливо для забезпечення задоволення користувачів та оптимізації ресурсів. Загальна ціль тестування полягає в тому, щоб забезпечити найвищу якість програмного засобу та зменшити ризик виникнення проблем під час експлуатації.

Тестування буде проведено з використанням двох сценаріїв:

– перевірка спроби впровадження SQL–ін'єкції;
– перевірка введення нормальних даних користувача.


Рисунок 3.7 – Вигляд додатку з введенням SQL–ін'єкції в поле Username

На рисунку 3.7 зображено фрагмент веб-сайту, розробленого для тестування розробленого программного засобу виявлення SQL–ін'єкції методами машинного навчання. Як бачимо, користувач намагається ввести шкідливий SQL–запит.

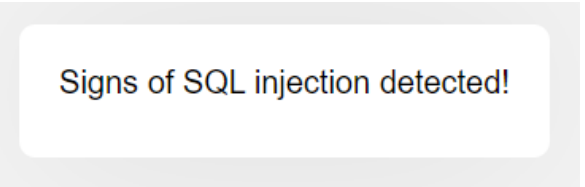



Рисунок 3.8 – Вигляд додатку з виявлення SQL–ін'єкції

У результаті запит визначається моделлю машинного навчання як небезпечний та користувач отримує відповідну відповідь. Запит записується до бази даних для подальшого розширенн дата сету, для перенавчання та удосконалення точності передбачень моделлю в майбутньому.

Розглянемо приклад, що відбувається, коли користувач хоче залогуватись на веб-сайті, використовуючи свої дані, які вже є збережені у базі даних (рис. 3.9).


Рисунок 3.9 – Вигляд додатку з введенням під час логування користувача

Як бачимо з рисунку 3.10 логування відбулось успішно. Запит є безпечним, та записується до бази даних, адже для подальшого збільшення дата сету необхідні як шкідливі так і нормальні приклади HTTP-запитів.

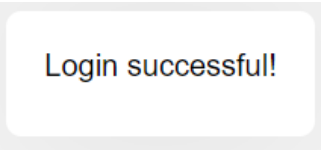



Рисунок 3.10 – Вигляд додатку при успішному логуванні

Засіб був протестований на контрольній вибірці даних, яка складалась з 21 сигнутари, серед яких 10 з SQL–ін'єкціями і 11 – безпечні.

Під час тестування модель допустила одну помилку. Було отримано високі показники точності, а саме 95%.

# 4 ЕКОНОМІЧНА ЧАСТИНА

## 4.1 Проведення комерційного та технологічного аудиту науково–технічної розробки

Науково–технічна розробка має право на існування та впровадження, якщо вона відповідає вимогам часу, як в напрямку науково–технічного прогресу та і в плані економіки. Тому для науково–дослідної роботи необхідно оцінювати економічну ефективність результатів виконаної роботи.

Магістерська кваліфікаційна робота на тему «Інформаційна технологія виявлення SQL–ін'єкцій методами машинного навчання» відноситься до науково–технічних робіт, які орієнтовані на виведення на ринок (або рішення про виведення науково–технічної розробки на ринок може бути прийнято у процесі проведення самої роботи), тобто коли відбувається так звана комерціалізація науково–технічної розробки. Цей напрямок є пріоритетним, оскільки результатами розробки можуть користуватися інші споживачі, отримуючи при цьому певний економічний ефект. Але для цього потрібно знайти потенційного інвестора, який би взявся за реалізацію цього проекту і переконати його в економічній доцільності такого кроку.

Для наведеного випадку нами мають бути виконані такі етапи робіт:

1) проведено комерційний аудит науково–технічної розробки, тобто встановлення її науково–технічного рівня та комерційного потенціалу;

2) розраховано витрати на здійснення науково–технічної розробки;

3) розрахована економічна ефективність науково–технічної розробки у випадку її впровадження і комерціалізації потенційним інвестором і проведено обґрунтування економічної доцільності комерціалізації потенційним інвестором.

Метою проведення комерційного і технологічного аудиту дослідження за темою «Інформаційна технологія виявлення SQL–ін'єкцій методами машинного

навчання» є оцінювання науково–технічного рівня та рівня комерційного потенціалу розробки, створеної в результаті науково–технічної діяльності.

Оцінювання науково–технічного рівня розробки та її комерційного потенціалу рекомендується здійснювати із застосуванням 5–ти бальної системи оцінювання за 12–ма критеріями, наведеними в табл. 4.1.

Таблиця 4.1 – Рекомендовані критерії оцінювання науково–технічного рівня і комерційного потенціалу розробки та бальна оцінка

| Бали (за 5–ти бальною шкалою) | | | | | |
|---|---|---|---|---|---|
| | 0 | 1 | 2 | 3 | 4 |
| Технічна здійсненність концепції | | | | | |
| 1 | Достовірність концепції не підтверджена | Концепція підтверджена експертними висновками | Концепція підтверджена розрахунками | Концепція перевірена на практиці | Перевірено працездатність продукту в реальних умовах |
| Ринкові переваги (недоліки) | | | | | |
| 2 | Багато аналогів на малому ринку | Мало аналогів на малому ринку | Кілька аналогів на великому ринку | Один аналог на великому ринку | Продукт не маєаналогів на великому ринку |
| 3 | Ціна продукту значно вища за ціни аналогів | Ціна продукту дещо вища за ціни аналогів | Ціна продукту приблизно дорівнює цінам аналогів | Ціна продукту дещо нижче за ціни аналогів | Ціна продукту значно нижче зацін аналогів |
| 4 | Технічні та споживчі властивості продукту значно гірші, ніж в | Технічні та споживчі влас–тивості продукту трохи гірші, ніж в аналогів | Технічні та споживчі властивості продукту на рівні аналогів | Технічні та споживчі властивості продукту трохи кращі, ніж в | Технічні та споживчі властивості продукту значно кращі, ніж в |
| 5 | Експлуатаційні витрати значно вищі, ніж в аналогів | Експлуатаційні витрати дещо вищі, ніж в аналогів | Експлуатаційні витрати на рівні експлуатаційних витрат аналогів | Експлуатаційні витрати трохи нижчі, ніж в аналогів | Експлуатацій нівитрати значнонижчі, ніж в аналогів |
| Ринкові перспективи | | | | | |
| 6 | Ринок малий і не має позитивної динаміки | Ринок малий, але має позитивну динаміку | Середній ринок з позитивною динамікою | Великий стабільний ринок | Великий ринок з позитивною динамікою |
| 7 | Активна конкуренція великих компаній на | Активна конкуренція | Помірна конкуренція | Незначна конкуренція | Конкурентів немає |

Продовження таблиці 4.1

| Практична здійсненність | | | | | |
|---|---|---|---|---|---|
| 8 | Відсутні фахівці як з технічної, так і з комерційної реалізації ідеї | Необхідно наймати фахівців або витрачати значні кошти та час на навчання наявних фахівців | Необхідне незначне навчання фахівців та збільшення їх штату | Необхідне незначне навчання фахівців | Є фахівці з питань як з технічної, так і з комерційної реалізації ідеї |
| 9 | Потрібні значні фінансові ресурси, які відсутні. Джерела фінансування ідеївідсутні | Потрібні незначні фінансові ресурси. Джерела фінансування відсутні | Потрібні значні фінансові ресурси. Джерела фінансування є | Потрібні незначні фінансові ресурси. Джерела фінансування є | Не потребує додаткового фінансування |
| 10 | Необхідна розробка нових матеріалів | Потрібні матеріали, що використовують ся у військово промисловому комплексі | Потрібні дорогі матеріали | Потрібні досяжні та дешеві матеріали | Всі матеріали дляреалізації ідеї відомі та давно використову ються у виробництві |
| 11 | Термін реалізації ідеї більший за 10 років | Термін реалізації ідеї більший за 5 років. Термін окупності інвестицій більше 10–ти років | Термін реалізації ідеї від 3–х до 5–ти років. Термін окупності інвестицій більше 5–ти років | Термін реалізації ідеї менше 3–х років. Термін окупності інвестицій від 3–хдо 5–ти років | Термін реалізаціїідеї менше 3–х років. Термін окупності інвестицій менше3–х років |
| 12 | Необхідна розробка регламентних документів та отримання великої кількості дозвільних документів на виробництво та реалізацію продукту | Необхідно отримання великої кількості дозвільних документів на виробництво та реалізацію продукту, що вимагає значних коштів та часу | Процедура отримання дозвільних документів для виробництва та реалізації продукту вимагає незначних коштів та часу | Необхідно тільки повідомлення відповідним органам про виробництво та реалізацію продукту | Відсутні будь– які регламентні обмеження на виробництво та реалізацію продукту |

Для проведення технологічного аудиту залучені 3 незалежних експерти:

– Керівник магістерської кваліфікаційної роботи, к. т. н., доцент кафедри захисту інформації Вінницького національного технічного університету Куперштейн Л. М.;

–к. т. н., доцент кафедри захисту інформації Вінницького національного технічного університету Баришев Ю. В.;

–к. т. н., доцент кафедри захисту інформації Вінницького національного технічного університету Войтович О. П.

Результати оцінювання науково–технічного рівня та комерційного потенціалу науково–технічної розробки потрібно звести до таблиці (табл. 4.2).

Таблиця 4.2 – Результати оцінювання науково–технічного рівня і комерційного потенціалу розробки експертами

| Критерії | Експерт (ПІБ, посада) | | |
|---|---|---|---|
| | Куперштейн Л. М. | Баришев Ю. В. | Войтович О. П. |
| | Бали: | | |
| 1. Технічна здійсненність концепції | 4 | 3 | 4 |
| 2. Ринкові переваги (наявність аналогів) | 3 | 2 | 3 |
| 3. Ринкові переваги (ціна продукту) | 3 | 3 | 3 |
| 4. Ринкові переваги (технічні властивості) | 2 | 3 | 2 |
| 5. Ринкові переваги (експлуатаційні витрати) | 3 | 3 | 2 |
| 6. Ринкові перспективи (розмір ринку) | 2 | 2 | 3 |
| 7. Ринкові перспективи (конкуренція) | 3 | 2 | 3 |
| 8. Практична здійсненність (наявність фахівців) | 5 | 5 | 5 |
| 9. Практична здійсненність (наявність фінансів) | 4 | 4 | 3 |
| 10. Практична здійсненність (необхідність нових матеріалів) | 4 | 5 | 5 |
| 11. Практична здійсненність (термін реалізації) | 4 | 4 | 3 |
| 12. Практична здійсненність (розробка документів) | 4 | 3 | 4 |
| Сума балів | 41 | 39 | 40 |
| Середньоарифметична сума балів $СБ_c$ | 40 | | |

За результатами розрахунків, наведених в таблиці 4.2, зробимо висновок щодо науково–технічного рівня і рівня комерційного потенціалу розробки. При цьому використаємо рекомендації, наведені в табл. 4.3.

Таблиця 4.3 – Науково–технічні рівні та комерційні потенціали розробки

| Середньоарифметична сума балів СБ ,розрахована на основі висновків експертів | Науково–технічний рівень та комерційнийпотенціал розробки |
|---|---|

| | |
|---|---|
| 41…48 | Високий |
| 31…40 | Вище середнього |
| 21…30 | Середній |
| 11…20 | Нижче середнього |
| 0…10 | Низький |

Згідно проведених досліджень рівень комерційного потенціалу розробки за темою «Інформаційна технологія виявлення SQL–ін'єкцій методами машинного навчання» становить 40 балів, що, відповідно до таблиці 4.3, свідчить про комерційну важливість проведення даних досліджень (рівень комерційного потенціалу розробки вище середнього).

## 4.2 Розрахунок узагальненого коефіцієнта якості розробки

Окрім комерційного аудиту розробки доцільно також розглянути технічний рівень якості розробки, розглянувши її основні технічні показники. Ці показники по–різному впливають на загальну якість проектної розробки.

Узагальнений коефіцієнт якості ($B_н$) для нового технічного рішення розрахуємо за формулою:

$$B_н = \sum_{i=1}^{k} \alpha_i \cdot \beta_i \quad (4.1)$$

де k – кількість найбільш важливих технічних показників, які впливають на якість нового технічного рішення;

$\alpha_i$ – коефіцієнт, який враховує питому вагу і–го технічного показника в загальній якості розробки. Коефіцієнт αі визначається експертним шляхом і при цьому має виконуватись умова

$$\sum_{i=1}^{k} \alpha_i = 1$$

$\beta_i$ – відносне значення *і–го* технічного показника якості нової розробки.

Відносні значення $\beta_i$ для різних випадків розраховуємо за такими формулами:

для показників, зростання яких вказує на підвищення в лінійній залежності якості нової розробки:

$$\beta_i = \frac{I_{ni}}{I_{ai}} \quad (4.2)$$

де $I_{ni}$ та $I_{ai}$ – чисельні значення конкретного і–го технічного показника якості відповідно для нової розробки та аналога;

–для показників, зростання яких $I_{ai}$ вказує на погіршення в лінійній залежності якості нової розробки:

$$\beta_i = \frac{I_{ni}}{} \quad (4.3)$$

Використовуючи наведені залежності можемо проаналізувати та порівняти техніко–економічні характеристики аналогу та розробки на основі отриманих наявних та проектних показників, а результати порівняння зведемо до таблиці 4.4.

Таблиця 4.4 – Порівняння основних параметрів розробки та аналога.

| Показники (параметри) | Одиниця вимірю–вання | Аналог | Проектованийпристрій | Відношення параметрів нової розробки до аналога | Питома вага показника |
|---|---|---|---|---|---|
| Достовірність визначення SQL–ін'єкцій | % | 80 | 95 | 1,19 | 0,3 |
| Швидкість визначення SQL–ін'єкцій | мс | 6 | 2 | 3 | 0,2 |
| Рівень захищеності даних | % | 87 | 95 | 1,09 | 0,2 |
| Універсальність | бал | 6 | 8 | 1,33 | 0,15 |
| Доступність інтерфейсу користувача | бал | 7 | 8 | 1,14 | 0,15 |

Узагальнений коефіцієнт якості ($B_н$) для нового технічного рішення складе:

$$B_н = \sum_{k}^{k} \alpha_i \cdot \beta_i = 1{,}19 \cdot 0{,}3 + 3 \cdot 0{,}2 + 1{,}09 \cdot 0{,}2 + 1{,}33 \cdot 0{,}15 + 1{,}14 \cdot 0{,}15 = 1{,}55$$

Отже за технічними параметрами, згідно узагальненого коефіцієнту якості розробки, науково–технічна розробка переважає існуючі аналоги приблизно в 1,55 рази.

## 4.3 Розрахунок витрат на проведення науково–дослідної роботи

Витрати, пов’язані з проведенням науково–дослідної роботи на тему «Інформаційна технологія виявлення SQL–ін'єкцій методами машинного навчання», під час планування, обліку і калькулювання собівартості науково–дослідної роботи групуємо за відповідними статтями.

4.3.1 Витрати на оплату праці

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

Основна заробітна плата дослідників

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

$$З_о = \sum_{i=1}^{k}$$

(4.4)

де $k$ – кількість посад дослідників залучених до процесу досліджень;

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

$t_i$ – число днів роботи конкретного дослідника, дн.;

$Т_р$ – середнє число робочих днів в місяці, $Т_р$ =21 дні.

$З_о$ = 20300,00 · 30 / 21 = 29000,00 грн.

Проведені розрахунки зведемо до таблиці 4.5.

Таблиця 4.5 – Витрати на заробітну плату дослідників

| Найменування посади | Місячний посадовий оклад, грн | Оплата за р | Число днів роботи | Витрати на заробітну плату, грн |
|---|---|---|---|---|
| Керівник проекту | 20300,00 | 966,67 | 30 | 29000,00 |
| Інженер–розробник програмного забезпечення | 18500,00 | 880,95 | 21 | 18500,00 |
| Науковий консультант з проблем машинного навчання | 17100,00 | 814,29 | 10 | 8143,00 |
| Всього | | | | 55644,00 |

Основна заробітна плата робітників

Витрати на основну заробітну плату робітників ($З_р$) за відповідними найменуваннями робіт НДР на тему «Інформаційна технологія виявлення SQL–ін'єкцій методами машинного навчання» розраховуємо за формулою:

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

(4.5)

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

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

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

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

(4.6)

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

$K_i$ – коефіцієнт міжкваліфікаційного співвідношення для встановлення тарифної ставки робітнику відповідного розряду;

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

$T_p$ – середнє число робочих днів в місяці, приблизно Тр = 21 дн;

$t_{зм}$ – тривалість зміни, год.

$C_1 = 6700{,}00 \cdot 1{,}10 \cdot 1{,}65 / (21 \cdot 8) = 72{,}38$ грн.

$З_{р1} = 72{,}38 \cdot 6{,}00 = 434{,}30$ *грн.*.

Таблиця 4.6 – Величина витрат на основну заробітну плату робітників

| Найменування робіт | Тривалість роботи, год | Розряд роботи | Тарифний коефіцієнт | Погодинна тарифна ставка, грн | Величина оплати на робітника грн |
|---|---|---|---|---|---|
| Підготовка робочого місця інженера– розробника ПЗ | 6,00 | 2 | 1,10 | 72,38 | 434,30 |
| Інсталяція програмного забезпечення середовища розробки | 5,10 | 3 | 1,35 | 88,83 | 453,06 |
| Формування кодів програмних блоків | 7,50 | 5 | 1,70 | 111,87 | 839,00 |
| Формування бази даних для забезпечення машинного навчання | 20,00 | 2 | 1,10 | 72,38 | 1447,60 |
| Контроль проходження експериментів | 9,00 | 4 | 1,50 | 98,71 | 888,39 |
| Всього | | | | | 4062,35 |

Додаткова заробітна плата дослідників та робітників

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

$$\frac{H_{дод}}{100\%}$$

$$З_{дод} = (З_о + З_р) \cdot \quad (4.7)$$

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

*$З_{дод}$ = (55644,00+ 4062,35) · 11 / 100% = 6567,70 грн.*

4.3.2 Відрахування на соціальні заходи

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

$$З_н = (З_о + З_р + З_{дод}) \cdot \frac{Н_{зп}}{100\%} \quad (4.8)$$

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

*Зн = (55644,00 + 4062,35 + 6567,70)* · 22 / 100% = 14580,29 грн.

4.3.3 Сировина та матеріали

До статті «Сировина та матеріали» належать витрати на сировину, основні та допоміжні матеріали, інструменти, пристрої та інші засоби і предмети праці, які придбані у сторонніх підприємств, установ і організацій та витрачені на проведення досліджень за темою «Інформаційна технологія виявлення SQL–ін'єкцій методами машинного навчання».

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

$$M = \sum_{j=1}^{n} Н_j \cdot Ц_j \cdot К_j - \sum_{j=1}^{n} В_j \cdot Ц_{в\,j} \quad (4.9)$$

де $Н_j$ – норма витрат матеріалу *j*–го найменування, кг;

*n* – кількість видів матеріалів;

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

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

$В_j$ – маса відходів *j*–го найменування, кг;

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

*$М_1$* = 4,0· 260,00 · *1,1 – 0 · 0* = 1040,00 *грн*

Проведені розрахунки зведемо до таблиці 4.7.

Таблиця 4.7 – Витрати на матеріали

| Найменування матеріалу, марка, тип, сорт | Ціна за 1 кг,грн | Норма витрат, кг | Величина відходів, кг | Ціна відходів, грн/кг | Вартість витраченого матеріалу, грн |
|---|---|---|---|---|---|
| Папір для принтера А4 Zoom | 260,00 | 4,0 | 0 | 0 | 1040,00 |
| Підставка–органайзер для формату А5, олівців, ручок, фломастерів (лхдф) на 7 відділень | 449,00 | 3,0 | 0 | 0 | 1347,00 |
| Набір кулькових ручок 555–А 50 шт | 93,41 | 1,0 | 0 | 0 | 93,41 |
| USB флеш накопичувач Transcend 64Gb JetFlash 700 (TS64GJF700) | 230,00 | 3,0 | 0 | 0 | 690,00 |
| Картридж для HP DeskJet 2050A HP 122 Black CH561HE | 950,99 | 1,0 | 0 | 0 | 950,99 |
| Тонер IPM HP LJ P1005 (TSH87B) black | 520,00 | 1,0 | 0 | 0 | 520,00 |
| Диск оптичний CD–RW | 23,00 | 2,0 | 0 | 0 | 50,60 |
| Всього | | | | | 4692,00 |

4.3.4 Розрахунок витрат на комплектуючі

Витрати на комплектуючі *($К_в$)*, які використовують при проведенні НДР на тему «Інформаційна технологія виявлення SQL–ін'єкцій методами машинного навчання», розраховуємо, згідно з їхньою номенклатурою, за формулою:

$$К_в = \sum_{j=1}^{n} Н_j \cdot Ц_j \cdot К_j \qquad (4.10)$$

де $Н_j$ – кількість комплектуючих $j$–го виду, шт.;

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

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

$К_{в1}$ = *1* · 50,00 · *1,11* = 55,50 *грн.*

Проведені розрахунки зведемо до таблиці 4.8.

Таблиця 4.8 – Витрати на комплектуючі

| Найменування комплектуючих | Кількість, шт. | Ціна за штуку, грн | Сума, грн |
|---|---|---|---|
| Кабель для передачі даних SATA III 1.0m Cablexpert(CC–SATAM–DATA–XL) | 1 | 50,00 | 55,50 |
| Wi–Fi адаптер TP–LINK TL–WN725N | 1 | 299,00 | 328,9 |
| Всього | | | 2536,69 |

4.3.5 Спецустаткування для наукових (експериментальних) робіт

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

Балансову вартість спецустаткування розраховуємо за формулою:

$$B_{спец} = \sum_{j=1}^{k} Ц_i \cdot C_{пр.i} \cdot K_i \qquad (4.11)$$

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

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

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

$k$ – кількість найменувань устаткування.

$B_{спец1}$ = *1345,00· 2 · 1,1 = 2959,00грн.*

Отримані результати зведемо до таблиці 2.9:

Таблиця 4.9 – Витрати на придбання спецустаткування по кожному виду

| Найменування устаткування | Кількість, шт | Ціна за одиницю, грн | Вартість, грн |
|---|---|---|---|
| Оперативна пам'ять Kingston Fury DDR4–2666 8192 MB PC4–21300 | 2 | 1345,00 | 2959,00 |
| Диск HGST Travelstar 2.5–Inch 1TB | 1 | 5600,00 | 6160,00 |
| Відеокарта AMD Radeon HD 7670M | 1 | 7835,00 | 8618,50 |
| Всього | | | 17737,5 |

4.3.6 Програмне забезпечення для наукових (експериментальних) робіт

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

Балансову вартість програмного забезпечення розраховуємо за формулою:

$$B_{прг} = \sum_{j=1}^{k} Ц_{iпрг} \cdot C_{прг.i} \cdot K_i \qquad (4.12)$$

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

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

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

$k$ – кількість найменувань програмних засобів.

$B_{прг1} = 5830,00 \cdot 1 \cdot 1,1 = 6413,00грн.$

Отримані результати зведемо до таблиці 4.10:

Таблиця 4.10 – Витрати на придбання програмних засобів по кожному виду

| Найменування програмного засобу | Кількість, шт | Ціна за одиницю, грн | Вартість, грн |
|---|---|---|---|
| Прикладне програмне забезпечення розробки | 1 | 5830,00 | 6413,00 |
| Середовище програмування PyCharm | 1 | 850,00 | 935,00 |
| Всього | | | 7348,00 |

4.3.7 Амортизація обладнання, програмних засобів та приміщень

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

$$A_{обл} = \frac{Ц_б}{T_в} \cdot \frac{t_{вик}}{12} \quad (4.13)$$

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

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

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

$A_{обл1} = (57999{,}00 \cdot 2) / (2 \cdot 12) = 4833{,}25 грн.$

Проведені розрахунки зведемо до таблиці 4.11.

Таблиця 4.11 – Амортизаційні відрахування по кожному виду обладнання

| Найменування обладнання | Балансова вартість, грн | Строк корисного використання, років | Термін використання обладнання, місяців | Амортизаційні відрахування, грн |
|---|---|---|---|---|
| Ноутбук Asus ROG Strix G614JU–N4224 | 57999,00 | 2 | 2 | 4833,25 |
| Робоче місце інженера–розробника ПЗ | 8700,00 | 5 | 2 | 290,00 |
| Пристрої передачі даних комутатор мережевий TP–Link | 2780,00 | 2 | 2 | 231,67 |
| ОС Windows 11 | 8570,00 | 2 | 2 | 714,17 |
| Прикладний пакет Microsoft Office 2019 | 7825,00 | 2 | 2 | 652,08 |
| Оргтехніка | 7250,00 | 4 | 2 | 302,08 |
| Приміщення лабораторії | 675200,00 | 25 | 2 | 4501,33 |
| Всього | | | | 11524,58 |

4.3.8 Паливо та енергія для науково–виробничих цілей

Витрати на силову електроенергію (Ве) розраховуємо за формулою:

$$B_e = \sum_{j=1}^{n} \frac{W_{yi} \cdot t_i \cdot Ц_e \cdot K_{вni}}{\eta_i} \quad (4.14)$$

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

$t_i$ – тривалість роботи обладнання на етапі дослідження, год;

$Ц_e$ – вартість 1 кВт–години електроенергії, грн; (вартість електроенергії визначається за даними енергопостачальної компанії), приймемо Це = 6,50 грн;

$K_{вni}$ – коефіцієнт, що враховує використання потужності, $K_{вni}$ <1;

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

$B_{e1} = 0,30 \cdot 170,0 \cdot 6,50 \cdot 0,95 / 0,97 = 324,66$ *грн.*

Проведені розрахунки зведемо до таблиці.

Таблиця 4.12 – Витрати на електроенергію

| Найменування обладнання | Встановлена потужність, кВт | Тривалість роботи, год | Сума, грн |
|---|---|---|---|
| Ноутбук для проведення розробкиAsus ROG Strix G614JU–N4224 | 0,30 | 170,0 | 324,66 |
| Робоче місце інженера–розробника ПЗ | 0,12 | 170,0 | 122,19 |
| Пристрої передачі даних | 0,01 | 150,0 | 9,55 |
| Комутатор мережевий TP–Link TL–SG1005D | 0,01 | 140,0 | 8,91 |
| Оргтехніка | 0,50 | 3,0 | 9,54 |
| Всього | | | 474,80 |

4.3.9 Службові відрядження

До статті «Службові відрядження» дослідної роботи на тему «Інформаційна технологія виявлення SQL–ін'єкцій методами машинного навчання» належать витрати на відрядження штатних працівників, працівників організацій, які працюють за договорами цивільно– правового характеру,

аспірантів, зайнятих розробленням досліджень, відрядження, пов'язані з проведенням випробувань машин та приладів, а також витрати на відрядження на наукові з'їзди, конференції, наради, пов'язані з виконанням конкретних досліджень.

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

$$B_{св} = (З_о + З_р) \cdot \frac{Н_{св}}{100\%} \quad (4.15)$$

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

*$B_{св}$ = (55644,00 + 4062,35) · 20 / 100% = 11941,27 грн.*

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

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

$$B_{сп} = (З_о + З_р) \cdot \frac{Н_{сп}}{100\%} \quad (4.16)$$

де $Н_{сп}$ – норма нарахування за статтею «Витрати на роботи, які виконують сторонні підприємства, установи і організації», приймемо $Н_{сп}$= 35%.

*$B_{сп}$ = (55644,00 + 4062,35) · 35 / 100% = 20897,22 грн.*

4.3.11 Інші витрати

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

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

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

(4.17)

де $Н_{ів}$ – норма нарахування за статтею «Інші витрати», приймемо $Н_{ів}$ = 70%.

*$І_в$ = (55644,00 + 4062,35)· 70 / 100% = 41794,45 грн.*

4.3.12 Накладні (загальновиробничі) витрати

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

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

$$В_{нзв} = (З_о + З_р) \cdot$$

(4.18)

де $Н_{нзв}$ – норма нарахування за статтею «Накладні (загальновиробничі) витрати», приймемо $Н_{нзв}$ = 120%.

*$В_{нзв}$ = (55644,00 + 4062,35) · 120 / 100% = 71647,62 грн.*

Витрати на проведення науково–дослідної роботи на тему «Інформаційна технологія визначення операційної системи мережевих вузлів із застосуванням машинного навчання» розраховуємо як суму всіх попередніх статей витрат за формулою:

$$В_{заг} = З_о + З_р + З_{дод} + З_н + М + К_в + В_{спец} + В_{прг} + А_{обл} + В_е + В_{св} + В_{сп} + І_в + В_{нзв} \quad (4.18)$$

*$В_{заг}$=55644,00+4062,35+6567,70+14580,29+4692,00+2536,69+17737,50+734 8,00+11524,58+474,80+11941,27+20897,22+41794,45+71647,6 2=271 448,47*

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

$$ЗВ = \frac{В_{заг}}{\eta}$$

(4.19)

де η – коефіцієнт, який характеризує етап (стадію) виконання науково–дослідної роботи, приймемо η =0,90.

$$ЗВ = 271\ 448{,}47/0{,}90 = 301609{,}41$$

## 4.4 Розрахунок економічної ефективності науково–технічної розробки

В ринкових умовах узагальнюючим позитивним результатом, що його може отримати потенційний інвестор від можливого впровадження результатів тієї чи іншої науково–технічної розробки, є збільшення у потенційного інвестора величини чистого прибутку.

Результати дослідження проведені за темою «Інформаційна технологія виявлення SQL–ін'єкцій методами машинного навчання» передбачають комерціалізацію протягом 4–х років реалізації на ринку.

В цьому випадку основу майбутнього економічного ефекту будуть формувати:

$\Delta N$ – збільшення кількості споживачів продукту, у періоди часу, що

аналізуються, від покращення його певних характеристик;

Таблиця 4.13 – Збільшення кількості споживачів

| Показник | 1–й рік | 2–й рік | 3–й рік | 4–й рік |
|---|---|---|---|---|
| Збільшення кількості споживачів, осіб | 700 | 900 | 1000 | 500 |

$N$ – кількість споживачів які використовували аналогічний продукт у році до впровадження результатів нової науково–технічної розробки, приймемо 5000 осіб;

$Ц_б$ – вартість програмного продукту у році до впровадження результатів розробки, приймемо 1500,00 грн;

$\pm\Delta Ц_о$ – зміна вартості послуги від впровадження результатів, приймемо 250,50 грн.

Можливе збільшення чистого прибутку у потенційного інвестора Пі для кожного із 4–х років, протягом яких очікується отримання позитивних результатів від можливого впровадження та комерціалізації науково–технічної розробки, розраховуємо за формулою:

$$\Delta П_i = (\pm \Delta Ц_o \cdot N + Ц_o \cdot \Delta N)_i \cdot \lambda \cdot \rho \cdot (1 - \frac{\vartheta}{100}) \quad (4.20)$$

де $\lambda$ – коефіцієнт, який враховує сплату потенційним інвестором податку на додану вартість. У 2023 році ставка податку на додану вартість складає 20%, а коефіцієнт $\lambda = 0{,}8333$;

$\rho$ – коефіцієнт, який враховує рентабельність інноваційного продукту).

Приймемо $\rho = 0{,}33$

$\vartheta$ – ставка податку на прибуток, який має сплачувати потенційний інвестор, у 2023 році $\vartheta = 18\%$;

Збільшення чистого прибутку 1–го року:

$\Delta П_1 = (250{,}50 \cdot 5000{,}00 + 1750{,}50 \cdot 700) \cdot 0{,}83 \cdot 0{,}33 \cdot (1 – 0{,}18) = 556520{,}15$грн.

Збільшення чистого прибутку 2–го року:

$\Delta П_2 = (250{,}50 \cdot 5000{,}00 + 1750{,}50 \cdot 1600) \cdot 0{,}83 \cdot 0{,}33 \cdot (1 – 0{,}18) = 910363{,}07$грн.

Збільшення чистого прибутку 3–го року:

$\Delta П_3 = (250{,}50 \cdot 5000{,}00 + 1750{,}50 \cdot 2600) \cdot 0{,}83 \cdot 0{,}33 \cdot (1 – 0{,}18) = 1303521{,}87$грн.

Збільшення чистого прибутку 4–го року:

$\Delta П_4 = (250{,}50 \cdot 5000{,}00 + 1750{,}50 \cdot 3100) \cdot 0{,}83 \cdot 0{,}33 \cdot (1 – 0{,}18) = 1500101{,}27$ грн.

Приведена вартість збільшення всіх чистих прибутків *ПП*, що їх може отримати потенційний інвестор від можливого впровадження та комерціалізації науково–технічної розробки:

$$ПП = \sum_{i=1}^{T} \frac{\Delta П_i}{(1+\tau)^t} \quad (4.21)$$

де $\Delta П_i$ – збільшення чистого прибутку у кожному з років, протягом яких виявляються результати впровадження науково–технічної розробки, грн;

*Т* – період часу, протягом якого очікується отримання позитивних результатів від впровадження та комерціалізації науково–технічної розробки, роки;

$\tau$ – ставка дисконтування, за яку можна взяти щорічний прогнозований рівень інфляції в країні, $\tau$ =0,11;

*t* – період часу (в роках) від моменту початку впровадження науково–технічної розробки до моменту отримання потенційним інвестором додаткових чистих прибутків у цьому році.

$$ПП = 556520,15/(1+0,11)^1 + 910363,07/(1+0,11)^2 + 1303521,87/(1+0,11)^3 +$$
$$+1500101,27/(1+0,11)^4 = 501369,50 + 1119746,58 + 1785812,63 + 3720250,48 =$$
$$= 7\ 127\ 179,19$$

Величина початкових інвестицій *PV*, які потенційний інвестор має вкласти для впровадження і комерціалізації науково–технічної розробки:

$$PV = k_{інв} \cdot ЗВ, \tag{4.22}$$

Де $k_{інв}$ – коефіцієнт, що враховує витрати інвестора на впровадження науково–технічної розробки та її комерціалізацію, приймаємо $k_{інв}$ = 2,5;

*ЗВ* – загальні витрати на проведення науково–технічної розробки та оформлення її результатів, приймаємо *301609,41* грн.

$PV = k_{інв} \cdot ЗВ = 2,5 \cdot 301609,41 =$ 754023,525 грн..

Абсолютний економічний ефект $E_{абс}$ для потенційного інвестора від можливого впровадження та комерціалізації науково–технічно розробки становитиме:

$$E_{абс} = ПП - PV \tag{4.23}$$

де *ПП* – приведена вартість зростання всіх чистих прибутків від можливого впровадження та комерціалізації науково–технічної розробки, 7127179,19 грн;

$PV$ – теперішня вартість початкових інвестицій, 754023,525 грн.

Еабс = ПП – PV = 7127179,19 – 754023,525 = 6373155,67грн.

Внутрішня економічна дохідність інвестицій $E_в$, які можуть бути вкладені потенційним інвестором у впровадження та комерціалізацію науково–технічної розробки:

$$E_в = \sqrt[\text{ТЖ}]{1 + \frac{\text{Е}_{\text{абс}}}{PV}} - 1 \quad (4.24)$$

де $E_{абс}$ – абсолютний економічний ефект вкладених інвестицій, 22023858,38 грн;

$PV$ – теперішня вартість початкових інвестицій, 792222,40 грн;

$T_ж$ – життєвий цикл науково–технічної розробки, тобто час від початку її

Розробки до закінчення отримування позитивних результатів від її впровадження, 4 роки.

$$E_в = \sqrt[\text{ТЖ}]{1 + \frac{\text{Е}_{\text{абс}}}{PV}} - 1 = \left(1 + \frac{6373155{,}67}{754023{,}525}\right)^{1/4} - 1 = 0{,}75$$

Мінімальна внутрішня економічна дохідність вкладених інвестицій $\tau_{мін}$:

$$\tau_{мін} = d + f \quad (4.25)$$

де $d$ – середньозважена ставка за депозитними операціями в комерційних банках в 2023 році в Україні d =0,12;

$f$ – показник, що характеризує ризикованість вкладення інвестицій, приймемо 0,25.

$\tau_{мін}$ = 0,12+0,25 = 0,37 < 1,32 свідчить про те, що внутрішня економічна дохідність інвестицій $E_в$, які можуть бути вкладені потенційним інвестором у впровадження та комерціалізацію науково–технічної розробки вища мінімальної внутрішньої дохідності. Тобто інвестувати в науково–дослідну роботу за темою «Інформаційна технологія виявлення SQL–ін'єкцій методами машинного навчання» доцільно.

Період окупності інвестицій Ток які можуть бути вкладені потенційним інвестором у впровадження та комерціалізацію науково–технічної розробки:

$$T_{ок} = \frac{1}{E} \tag{4.26}$$

де $E_в$ – внутрішня економічна дохідність вкладених інвестицій.

$T_{ок}$ = 1 / 0,75 = 1,33 р.

$T_{ок}$ < 3–х років, що свідчить про комерційну привабливість науково–технічної розробки і може спонукати потенційного інвестора профінансувати впровадження даної розробки та виведення її на ринок.

Згідно проведених досліджень рівень комерційного потенціалу розробки за темою «Інформаційна технологія виявлення SQL–ін'єкцій методами машинного навчання» становить 40 балів, що, свідчить про комерційну важливість проведення даних досліджень (рівень комерційного потенціалу розробки вище середнього).

При оцінюванні за технічними параметрами, згідно узагальненого коефіцієнту якості розробки, науково–технічна розробка переважає існуючі аналоги приблизно в 1,55 рази.

Також термін окупності становить 1,33 р., що менше 3–х років, що свідчить про комерційну привабливість науково–технічної розробки і може спонукати потенційного інвестора профінансувати впровадження даної розробки та виведення її на ринок.

Отже можна зробити висновок про доцільність проведення науково–дослідної роботи за темою «Інформаційна технологія виявлення SQL–ін'єкцій методами машинного навчання».

# ВИСНОВКИ

У ході виконання магістерської кваліфікаційної роботи було виконано усі поставлені задачі. Було розглянуто, наскільки важливо в кібербезпеці вирішити проблему виявлення SQL–ін'єкцій, проведено дослідження існуючих SQL–ін'єкцій, розглянуто основні класи ін’єкцій SQL, їхні характеристики та проведено аналіз наявних комерційних методів і засобів їх виявлення. Проаналізовано програмні засоби виявлення SQL–ін’єкцій щодо того, наскільки швидко вони виконують перевірку. Як наслідок цього аналізу, було показано, що створення відповідного програмного забезпечення є необхідним. На основі проаналізованї інформації, були визначені функціональні та нефункціональні вимоги, а також цілі, які необхідно досягти під час процесу розробки.

Виконано розробку структури інформаційної технології виявлення SQL–ін'єкцій методами машинного навчання що включає у себе процес збору даних, процес попередньої обробки, процес інтелектуального аналізу та передбачення та процес до навчання чи перенавчання моделі.

Виконано обґрунтування вибору інструментальних засобів розробки. Результати включають мову програмування Python і середовище розробки Google Colab . Програмний засіб, який реалізує інформаційну виявлення SQL–ін'єкцій методами машинного навчання, розроблений на основі створених вимог до програмного засобу, архітектури системи та алгоритмів.

Розроблено інформаційну технологію, щоб покращити процес виявлення SQL–ін'єкцій методами машинного навчання. Для виконання даної задачі було проведено моделювання ряду класифікаторів. Спроектовано архітектуру моделі машинного навчання, відображено результати навчання у вигляді матриці помилок, оцінок навчання та кількості помилок першого та другого роду. На основі результатів було обрано класифікатор з найкращими показниками – CNN.

Розроблений програмний продукт пройшов тестування. Тестування проводилося за двома різними сценаріями роботи: з шкідливими та безпечними запитами. Користувачу надсилаються відповідні повідомлення для усунення проблем, а запити обробляються.

Результати дослідження щодо рівня комерційного потенціалу розробки показують, що проведення даних досліджень є комерційно важливим — рівень потенціалу розробки становить 40 балів, що є вище середнього. Згідно з оцінкою конкурентоспроможності, науково–технічна розробка перевершує існуючі аналоги приблизно в 1,55 рази. Крім того, термін окупності становить 1,33 року, що вказує на те, що науково–технічна розробка є комерційно привабливою. Результати показують, що наукові дослідження необхідні.

Даний програмний засіб є корисним застосунком для виявлення SQL–ін'єкцій методами машинного навчання.

# ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Тіщенко Д. С., Куперштейн Л. М. Міжнародна науково-практична Інтернет-конференція студентів, аспірантів та молодих науковців «Молодь в науці: дослідження, проблеми, перспективи (МН-2024)». : тези доповідей. Вінниця: ВНТУ, 2023. URL : https://conferences.vntu.edu.ua/index.php/mn/mn2024/paper/view /19781 (дата звернення 17.12.2023)

2. AWS Documantation URL: https://docs.aws.amazon.com/index.html?nc2=h_ql_doc (дата звернення: 05.11.2023).

3. Free CDN to Speed Up and Secure WebSite URL: https://geekflare.com/free–cdn–list/ (дата звернення: 05.11.2023).

4. OWASP CheatSheetSeries URL: https://github.com/OWASP/CheatSheetSeries/blob/master/cheatsheets/SQL_Inject ion_Prevention_Cheat_Sheet.md(дата звернення: 05.11.2023).

5. SQL Injections Top Attack Statistics URL: https://www.darkreading.com/risk/sql–injections–top–attack–statistics/d/d– id/1132988 (дата звернення: 05.11.2023).

6. Best Practices to prevent SQL–injections URL: https://tableplus.io/blog/2018/08/best–practices–to–prevent–sql– injection– attacks.html (дата звернення: 05.11.2023).

7. Akamai Report. The State of the Internet URL: https://www.akamai.com/us/en/resources/our–thinking/state–of– the–internet– report/global–state–of–the–internet–security–ddos–attack–reports.jsp (дата звернення: 05.11.2023).

8. Cloudflare Business Plan URL: https://www.cloudflare.com/plans/business/ (дата звернення: 05.11.2023).

9. Cloud Web Application Firewall URL: https://www.cloudflare.com/waf/ (дата звернення: 05.11.2023).

10. Libinjection URL: https://github.com/client9/libinjection(дата звернення: 05.11.2023).

11. SQL–injections: vulnerabilities and how to prevent attacks URL: https://www.veracode.com/security/sql–injection (дата звернення: 05.11.2023).

12. SQL Injection Prevention Cheat Sheet URL: https://goo.gl/N1NHtW. (дата звернення: 05.11.2023).

13. SQL Injection Attacks Are Rampant: How to Stop Your Next Hack Attack URL: https://goo.gl/RxnbWp. (дата звернення: 05.11.2023).

14. What are Convolutional Neural Networks?. *IBM*. URL: https://www.ibm.com/cloud/learn/convolutional–neural–networks (дата звернення: 05.10.2023).

15. Dang T. Guide to accuracy, precision, and recall. Mage.

URL: https://www.mage.ai/blog/definitive–guide–to–accuracy–precision–recall–for–product– developers (дата звернення: 08.11.2023).

16. GitHub: Where the world builds software. GitHub. URL: https://github.com/(дата звернення: 08.11.2023).

17. SQL ін’єкції в MySQL сервері URL: https://www.securitylab.ru/contest/212083.php (дата звернення: 08.11.2023).

18. SQL Injection Cheat Sheet URL: https://www.netsparker.com/blog/web–security/sql–injection–cheat–sheet/ (дата звернення: 08.11.2023).

19. Do stored procedures protect against SQL Injection? URL: https://blogs.msdn.microsoft.com/brian_swan/2011/02/16/do–stored–procedures–protect–against–sql–injection/ (дата звернення: 08.11.2023).

20. Simple guide to confusion matrix terminology: веб–сайт. URL: https://www.dataschool.io/simple–guide–to–confusion–matrix–terminology (дата звернення: 08.11.2023).

21. Open VAS – Open Vulnerability Assessment Scanner: веб–сайт. URL: https://www.openvas.org (дата звернення: 08.11.2023).

22. Testing for SQL Injection URL:

https://www.owasp.org/index.php/Testing
_for_SQL_Injection_(OTG–INPVAL–005) (дата звернення: 08.11.2023).

23. Gould C. JDBC Checker:A Static Analysis Tool for SQL/JDBC Applications / C. 24. Gould, Z. Su, P. Devanbu // In Proceedings of the 26th International Conference on Software Engineering (ICSE 04) Formal Demos, 2014. – pp 697-698. ) (дата звернення: 18.11.2023).

25. The Good and the Bad of Angular Development URL: https://goo.gl/GuZSXM. ) (дата звернення: 18.11.2023).

26. NoSQL URL: https://goo.gl/pYS5oz (дата звернення: 18.11.2023).

27. Cross-site scripting URL: https://goo.gl/Ja1Y14 (дата звернення: 18.11.2023).

28. Top 10 A1-Injection URL: https://goo.gl/comHQX (дата звернення: 08.11.2023).

# ДОДАТКИ

**Додаток А**

# ПРОТОКОЛ ПЕРЕВІРКИ МАГІСТЕРСЬКОЇ КВАЛІФІКАЦІЙНОЇ РОБОТИ НА НАЯВНІСТЬ ТЕКСТОВИХ ЗАПОЗИЧЕНЬ

Назва роботи: Інформаційна технологія виявлення SQL-ін'єкцій методами машинного навчання

Автор роботи: Тіщенко Дарина Сергіївна

Тип роботи: магістерська кваліфікаційна робота

Підрозділ кафедра захисту інформації ФІТКІ
(кафедра, факультет)

## Показники звіту подібності Unicheck

Оригінальність – 87,3 %. Схожість – 12,7 %.

Аналіз звіту подібності (відмітити потрібне):

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

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

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

Особа, відповідальна за перевірку _____ Валентина КАПЛУН
(підпис)

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

Автор роботи _____ Дарина ТІЩЕНКО
(підпис)

Керівник роботи _____ Леонід КУПЕРШТЕЙН

# Додаток Б
# Текст програми

***CNN.ipynb***

```
from google.colab import drive
drive.mount("/content/drive")

import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, classification_report,
confusion_matrix
from sklearn.preprocessing import LabelEncoder
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Embedding, Conv1D, GlobalMaxPooling1D,
Dense, Dropout
from tensorflow.keras.callbacks import EarlyStopping

df = pd.read_csv('/content/drive/MyDrive/SQL_injctions/SQL_Dataset.csv')

X = df['Query']
y = df['Label']

label_encoder = LabelEncoder()
y = label_encoder.fit_transform(y)

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)

max_words = 10000
tokenizer = Tokenizer(num_words=max_words)
tokenizer.fit_on_texts(X_train)
X_train_seq = tokenizer.texts_to_sequences(X_train)
X_test_seq = tokenizer.texts_to_sequences(X_test)

max_len = 100
X_train_pad = pad_sequences(X_train_seq, maxlen=max_len, padding='post')
X_test_pad = pad_sequences(X_test_seq, maxlen=max_len, padding='post')

model = Sequential()
model.add(Embedding(max_words, 50, input_length=max_len))
model.add(Conv1D(128, 5, activation='relu'))
model.add(GlobalMaxPooling1D())
model.add(Dense(64, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(1, activation='sigmoid'))

model.compile(optimizer='adam', loss='binary_crossentropy',
metrics=['accuracy'])
```

```
epochs = 10
batch_size = 32
history = model.fit(X_train_pad, y_train, epochs=epochs, batch_size=batch_size,
validation_split=0.2, callbacks=[EarlyStopping(monitor='val_loss', patience=2)])

model_config = model.get_config()
for layer in model_config['layers']:
    if 'config' in layer:
        print(f"\nLayer: {layer['config']['name']}")
        print("Config:")
        for key, value in layer['config'].items():
            print(f"  {key}: {value}")

y_pred = (model.predict(X_test_pad) > 0.5).astype("int32")

cm = confusion_matrix(y_test, y_pred)
print("Confusion Matrix:")
print(cm)

import matplotlib.pyplot as plt
import seaborn as sns

cm = confusion_matrix(y_test, y_pred)
plt.figure(figsize=(8, 6))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', xticklabels=['Non-Injection',
'Injection'], yticklabels=['Non-Injection', 'Injection'])
plt.title('Confusion Matrix for CNN')
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.show()

accuracy = accuracy_score(y_test, y_pred)
print("\nAccuracy:", accuracy)
print("\nClassification Report:")
print(classification_report(y_test, y_pred))

model.save('my_model.pkl')
```

***MySQL.sql***

```
CREATE DATABASE your_database;
USE your_database;

CREATE TABLE users (
    id INT AUTO_INCREMENT PRIMARY KEY,
    username VARCHAR(255) NOT NULL,
    password VARCHAR(255) NOT NULL
);
```

***index.html***

```
from flask import Flask, render_template, request, redirect, url_for
import mysql.connector

app = Flask(__name__)

app.config['MYSQL_HOST'] = 'localhost'
app.config['MYSQL_USER'] = 'root'
app.config['MYSQL_PASSWORD'] = ''
app.config['MYSQL_DB'] = 'sql_injection'
app.config['MYSQL_CURSORCLASS'] = 'DictCursor'

mysql = MySQL(app)

@app.route('/')
def index():
    return render_template('index.html')

@app.route('/login', methods=['POST'])
def login():
    username = request.form['username']
    password = request.form['password']


    query = "SELECT * FROM users WHERE username=%s AND password=%s"
    cursor.execute(query, (username, password))
    user = cursor.fetchone()


    load = open('/content/drive/MyDrive/SQL_injctions/my_model.pkl', 'rb')
    load_model = pickle.load(load)

    count_vect = CountVectorizer()
    df = pd.read_csv('/content/drive/MyDrive/SQL_injctions/sql.csv')
    count_vect.fit(df.text)

    check = pd.DataFrame({'text': [message.text]})
    check2 = count_vect.transform(check.text)
    load_model.predict(check2)


    if load_model.predict(check2) == 0:
        return "Login successful!"
    else:
        return "Signs of SQL injection detected"

if __name__ == '__main__':
    app.run(debug=True)
```

***style.css***

```
body {
    font-family: Arial, sans-serif;
    background-color: #f4f4f4;
    margin: 0;
    display: flex;
    justify-content: center;
    align-items: center;
    height: 100vh;
}

.container {
    background-color: #fff;
    padding: 20px;
    border-radius: 8px;
    box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
}

form {
    display: flex;
    flex-direction: column;
    align-items: center;
}

label {
    margin-bottom: 8px;
}

input {
    padding: 8px;
    margin-bottom: 16px;
}

button {
    padding: 10px;
    background-color: #053e83;
    color: #fff;
    border: none;
    border-radius: 40px;
    cursor: pointer;
}
```

***app.py***

```
from flask import Flask, render_template, request, redirect, url_for
import mysql.connector
```

```
app = Flask(__name__)

app.config['MYSQL_HOST'] = 'localhost'
app.config['MYSQL_USER'] = 'root'
app.config['MYSQL_PASSWORD'] = ''
app.config['MYSQL_DB'] = 'sql_injection'
app.config['MYSQL_CURSORCLASS'] = 'DictCursor'

mysql = MySQL(app)

@app.route('/')
def index():
    return render_template('index.html')

@app.route('/login', methods=['POST'])
def login():
    username = request.form['username']
    password = request.form['password']


    query = "SELECT * FROM users WHERE username=%s AND password=%s"
    cursor.execute(query, (username, password))
    user = cursor.fetchone()


    load = open('/content/drive/MyDrive/SQL_injctions/my_model.pkl', 'rb')
    load_model = pickle.load(load)

    count_vect = CountVectorizer()
    df = pd.read_csv('/content/drive/MyDrive/SQL_injctions/sql.csv')
    count_vect.fit(df.text)

    check = pd.DataFrame({'text': [message.text]})
    check2 = count_vect.transform(check.text)
    load_model.predict(check2)


    if load_model.predict(check2) == 0:
        return "Login successful!"
    else:
        return "Signs of SQL injection detected"

if __name__ == '__main__':
    app.run(debug=True)
```

***comparison.ipynb***

```
import pandas as pd
from sklearn.feature_extraction.text import CountVectorizer, TfidfVectorizer
from sklearn.datasets import fetch_20newsgroups
from sklearn.model_selection import train_test_split, GridSearchCV
```

```
from sklearn.feature_extraction.text import CountVectorizer, TfidfVectorizer
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score, classification_report

df = pd.read_csv('/content/drive/MyDrive/SQL_injctions/SQL_Dataset.csv')


X = df['Query']
y = df['Label']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)


count_vectorizer = CountVectorizer()
X = count_vectorizer.fit_transform(df['Query'])

dt_count_classifier = DecisionTreeClassifier(random_state=42)
dt_count_classifier.fit(X_train_count, y_train)
y_pred_count = dt_count_classifier.predict(X_test_count)

print("Results with CountVectorizer:")
print("Accuracy:", accuracy_score(y_test, y_pred_count))
print("Classification Report:\n", classification_report(y_test, y_pred_count))

tfidf_vectorizer = TfidfVectorizer()
X = tfidf_vectorizer.fit_transform(df['Query'])

dt_tfidf_classifier = DecisionTreeClassifier(random_state=42)
dt_tfidf_classifier.fit(X_train_tfidf, y_train)
y_pred_tfidf = dt_tfidf_classifier.predict(X_test_tfidf)

print("\nResults with TfidfVectorizer:")
print("Accuracy:", accuracy_score(y_test, y_pred_tfidf))
print("Classification Report:\n", classification_report(y_test, y_pred_tfidf))

print("CountVectorizer Matrix Shape:", X_train_count.shape)
print("TfidfVectorizer Matrix Shape:",  X_train_tfidf.shape)

# Перші 10 слів у словнику для CountVectorizer
print("\nCountVectorizer Vocabulary (First 10 words):",
list(count_vectorizer.vocabulary_.keys())[:10])

# Перші 10 слів у словнику для TfidfVectorizer
print("TfidfVectorizer Vocabulary (First 10 words):",
list(tfidf_vectorizer.vocabulary_.keys())[:10])
```

***DecisionTree.ipynb***

```
import pandas as pd
```

```
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score, classification_report,
confusion_matrix
import seaborn as sns
import matplotlib.pyplot as plt

df = pd.read_csv('/content/drive/MyDrive/SQL_injctions/SQL_Dataset.csv')

X = df['Query']
y = df['Label']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)

vectorizer = TfidfVectorizer()
X_train_vectorized = vectorizer.fit_transform(X_train)
X_test_vectorized = vectorizer.transform(X_test)

decision_tree_classifier = DecisionTreeClassifier()

param_grid = {'criterion': ['gini', 'entropy'], 'max_depth': [None, 10, 20, 30]}

grid_search = GridSearchCV(decision_tree_classifier, param_grid, cv=5,
scoring='accuracy')
grid_search.fit(X_train_vectorized, y_train)

print("Найкращі гіперпараметри:", grid_search.best_params_)

y_pred = grid_search.best_estimator_.predict(X_test_vectorized)

cm = confusion_matrix(y_test, y_pred)
plt.figure(figsize=(8, 6))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', xticklabels=['Non-Injection',
'Injection'], yticklabels=['Non-Injection', 'Injection'])
plt.title('Confusion Matrix for DecisionTree')
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.show()

accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)

print(classification_report(y_test, y_pred))
```

**Додаток В**

ІЛЮСТРАТИВНА ЧАСТИНА
ІНФОРМАЦІЙНА ТЕХНОЛОГІЯ ВИЯВЛЕННЯ SQL–ІН'ЄКЦІЙ МЕТОДАМИ МАШИННОГО НАВЧАННЯ

**СХЕМА ТИПІВ SQL–ІН'ЄКЦІЙ**

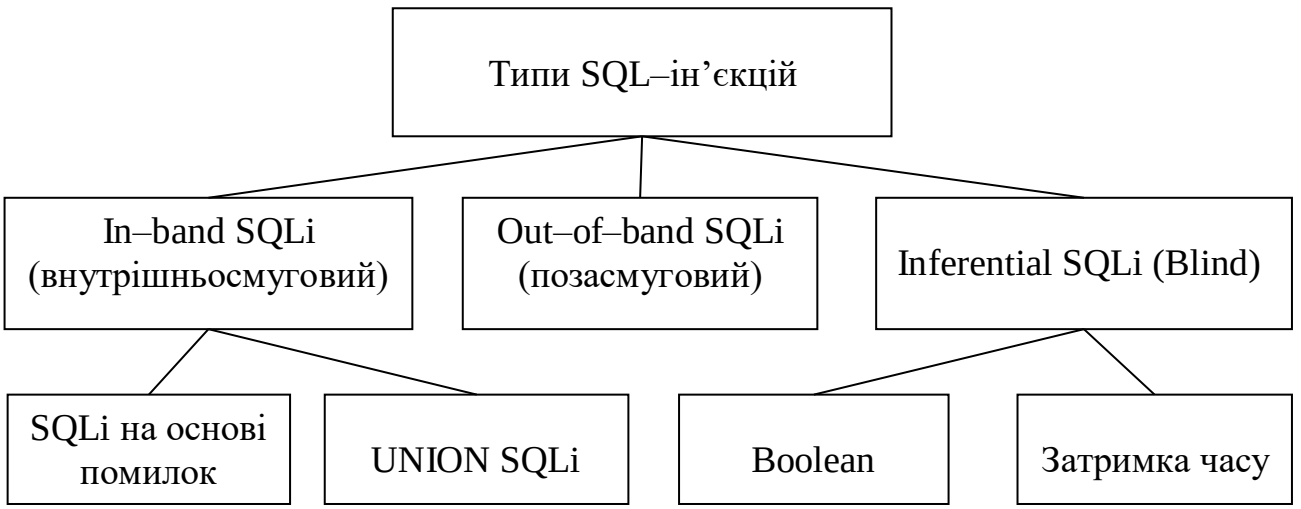


**СХЕМА ПРОЦЕСІВ ІНФОРМАЦІЙНОЇ ТЕХНОЛОГІЇ ВИЯВЛЕННЯ SQL–ІН'ЄКЦІЙ МЕТОДАМИ МАШИННОГО НАВЧАННЯ**

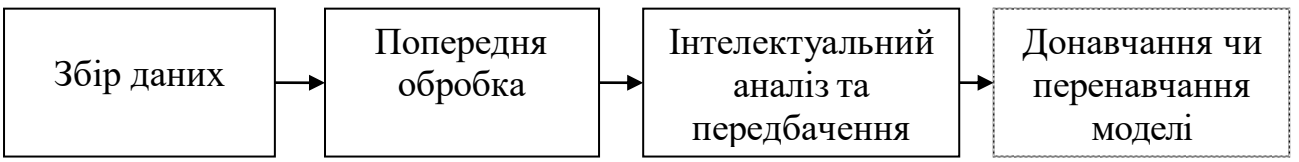


**СХЕМА ПРОЦЕСУ ЗБОРУ ДАНИХ**

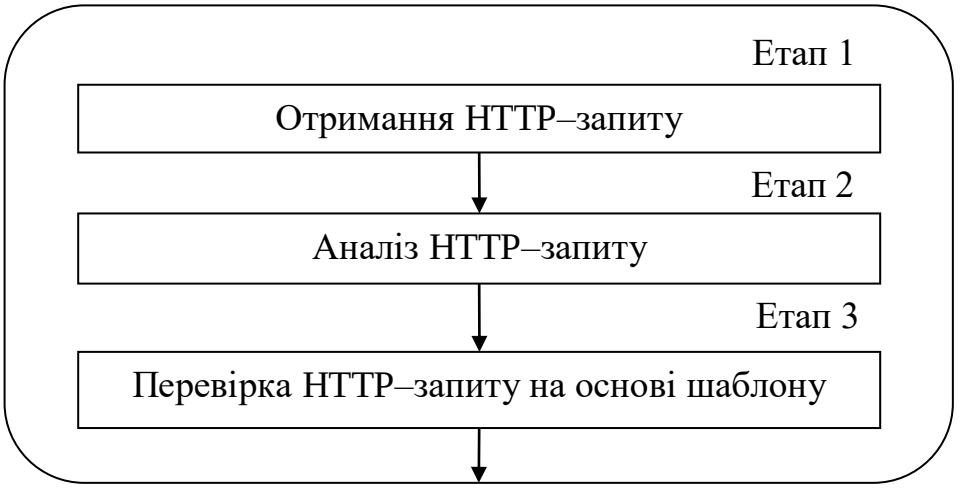


**СХЕМА ПРОЦЕСУ ПОПЕРЕДНЬОЇ ОБРОБКИ**

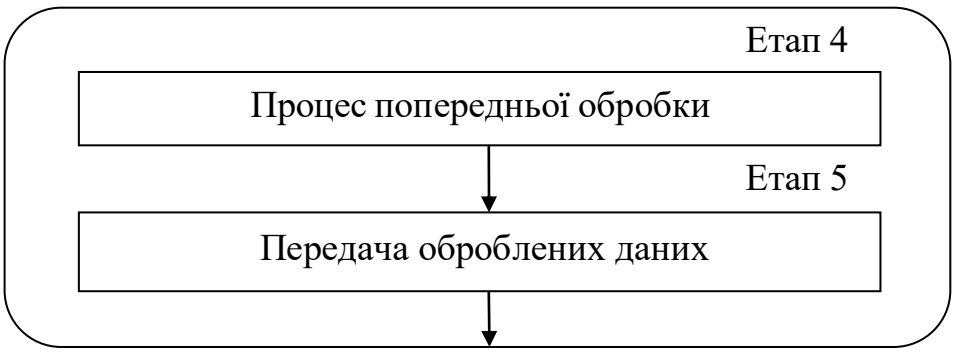
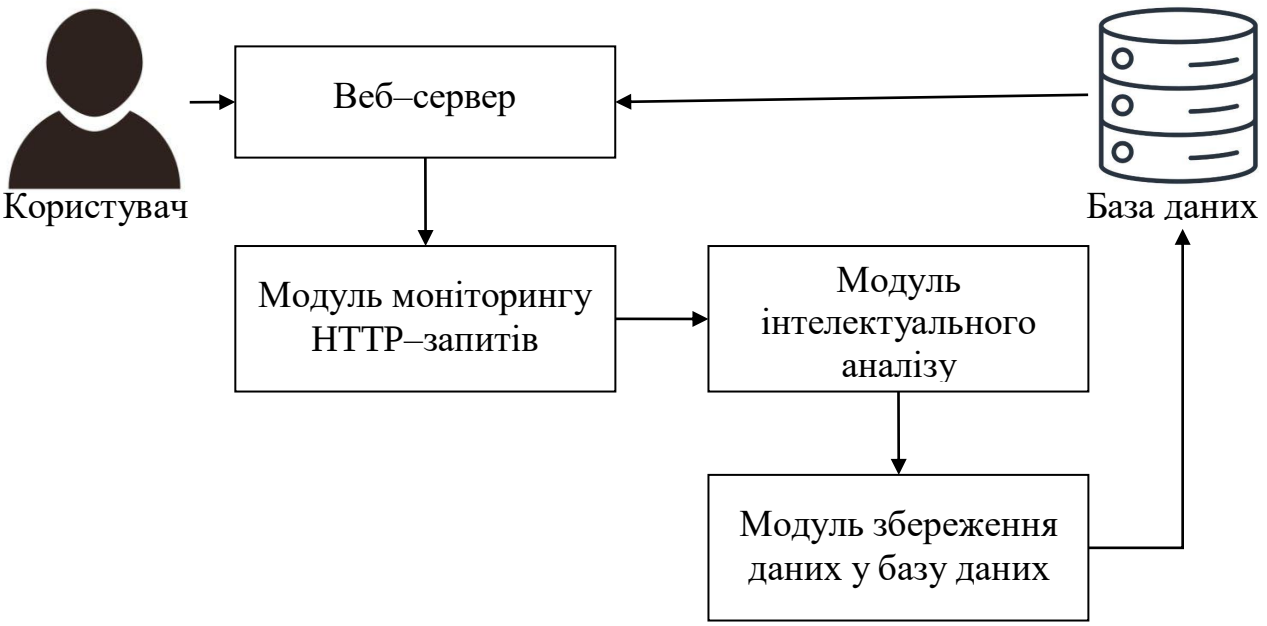


**СХЕМА ПРОЦЕСУ ІНТЕЛЕКТУАЛЬНОГО АНАЛІЗУ ТА ПЕРЕДБАЧЕННЯ**

**СХЕМА ПРОЦЕСУ ДОНАВЧАННЯ ЧИ ПЕРЕНАВЧАННЯ МОДЕЛІ**

**АРХТЕКТУРА ІНФОРМАЦІЙНОЇ ТЕХНОЛОГІЇ ВИЯВЛЕННЯ SQL–ІН'ЄКЦІЙ МЕТОДАМИ МАШИННОГО НАВЧАННЯ**



**АЛГОРИТМ РОБОТИ ІНФОРМАЦІЙНОЇ ТЕХНОЛОГІЇ ВИЯВЛЕННЯ SQL–ІН’ЄКЦІЙ МЕТОДАМИ МАШИННОГО НАВЧАННЯ**

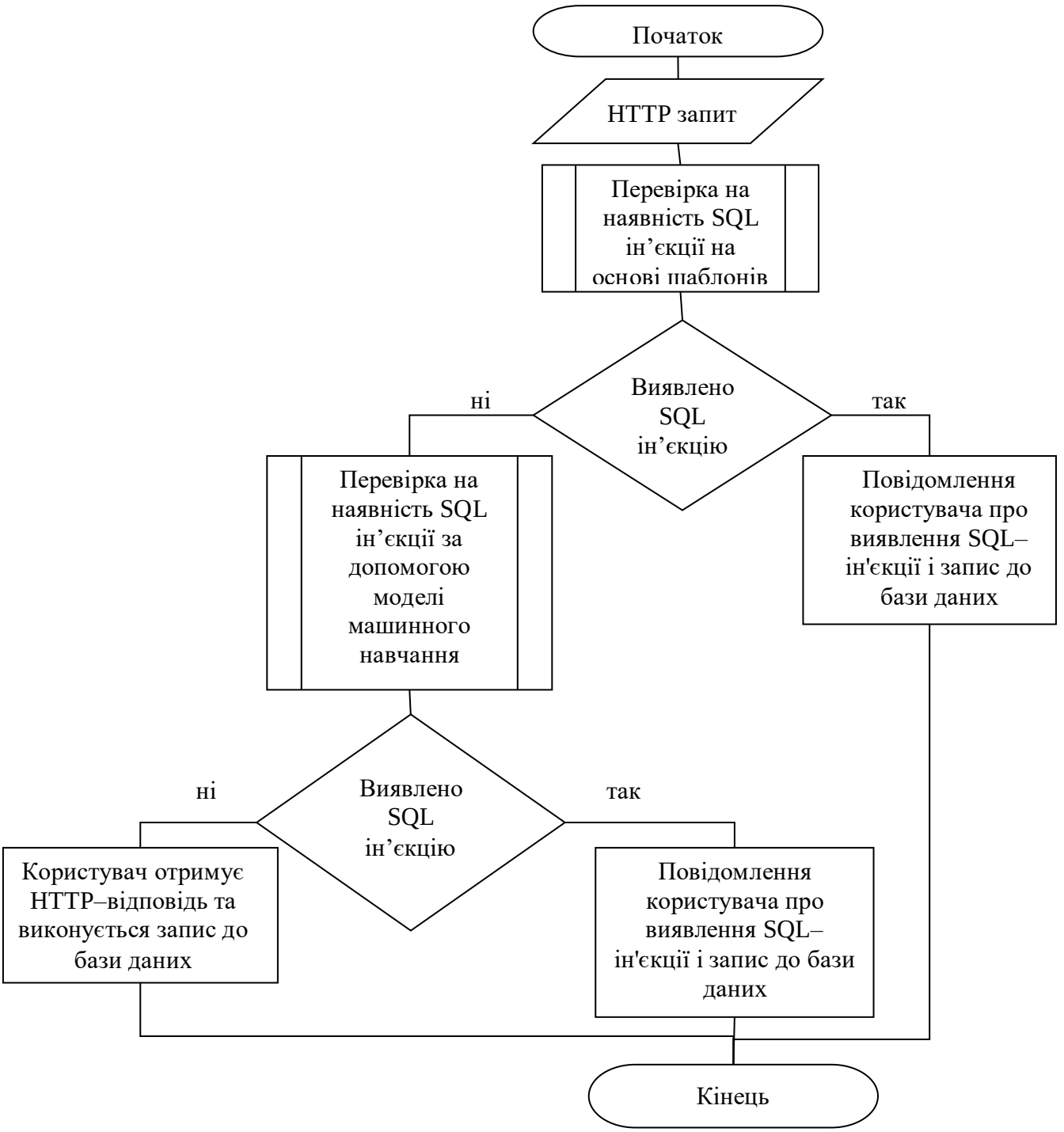
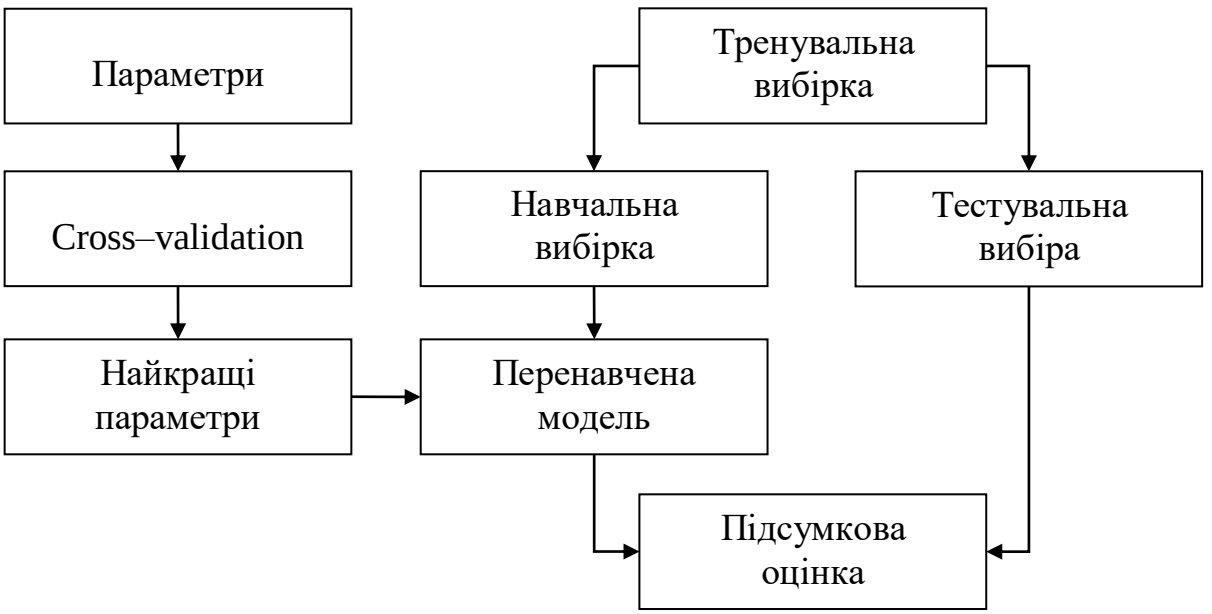


**СХЕМА ВАЛІДАЦІЙНОГО ПРОЦЕСУ**



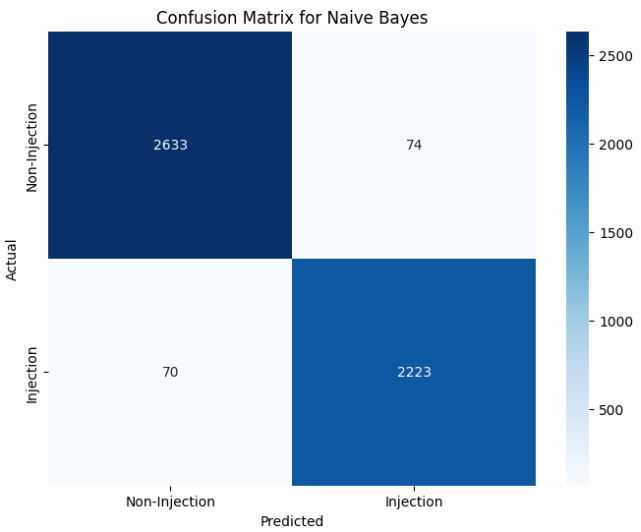
**ТАБЛИЦЯ РЕЗУЛЬТАТІВ МОДЕЛЮВАННЯ КЛАСИФІКАТОРІВ**

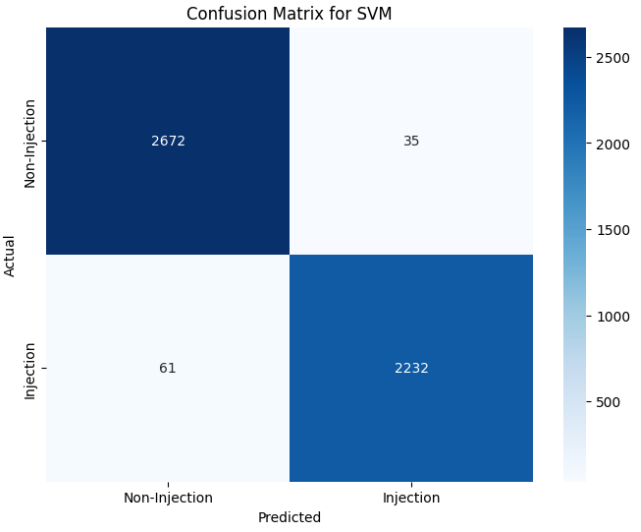
| Назва класифікатора | Accuracy | F1 | FP | FN | FPR | FNR |
|---|---|---|---|---|---|---|
| GaussianNB | 0,973 | 0,975 | 70 | 74 | 3,1% | 2,7% |
| SVM | 0,981 | 0,988 | 61 | 35 | 2,7% | 1,3% |
| KNearestNeighbors | 0,599 | 0,691 | 28 | 2021 | 1,2% | 76,7% |
| Logistic Regresion | 0,986 | 0,982 | 51 | 27 | 2,2% | 0,99% |
| DecisionTree | 0,983 | 0,987 | 79 | 9 | 3,4% | 0,33% |
| CNN | 0,991 | 0,991 | 35 | 3 | 1,5% | 0,1% |

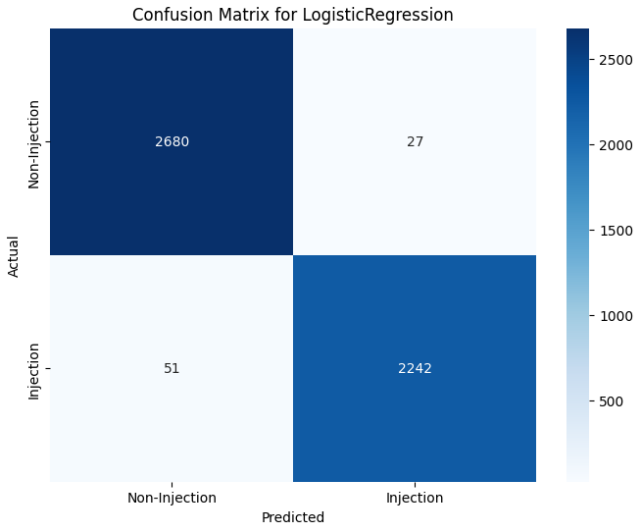
**ТАБЛИЦЯ АНАЛІЗУ ТЕХНОЛОГІЙ ВИЯВЛЕННЯ ТА ЗАПОБІГАННЯ SQL–ІН'ЄКЦІЯМ**

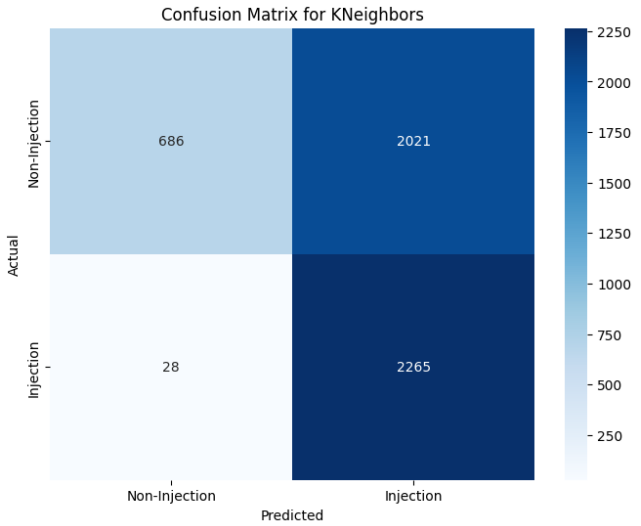
| № | Методи та засоби виявлення та протидії SQL–ін'єкціям | Blind SQL Injection | FF_SQLI | SQLI_XSS | SQLI_DNS | SQLI_CDP | SQLI_DDOS |
|---|---|---|---|---|---|---|---|
| 1. | Криптографічне хешування функцій | p | x | x | x | x | x |
| 2. | Динамічний перезапис файлів cookie | x | x | + | p | x | x |
| 3. | Статичний аналіз коду | o | x | o | x | * | x |
| 4. | Моніторинг часу роботи | p | + | * | x | x | x |
| 5. | Ardilla Tool | x | x | o | x | o | x |
| 6. | Noxes | x | x | p | x | + | x |
| 7. | Session Shield | x | x | * | p | x | x |
| 8. | AMNESIA | * | x | p | x | x | x |
| 9. | SQLMap | o | x | o | p | x | o |
| 10. | Fast Flux Monitor | x | o | x | o | x | x |

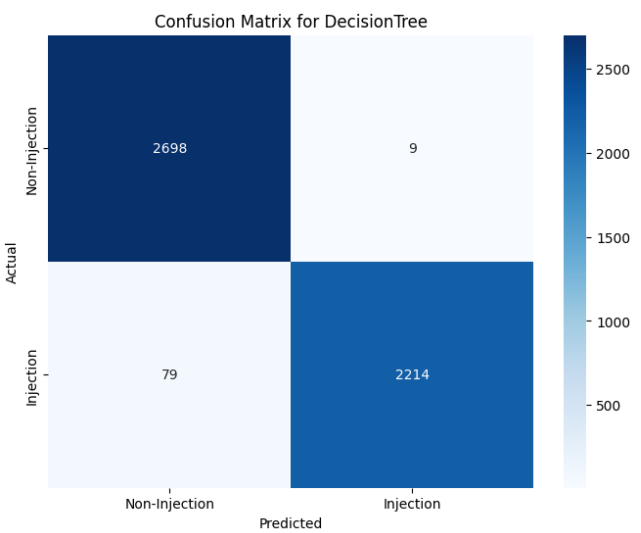
# МАТРИЦІ ПОМИЛОК ДЛЯ КЛАСИФІКАТОРІВ

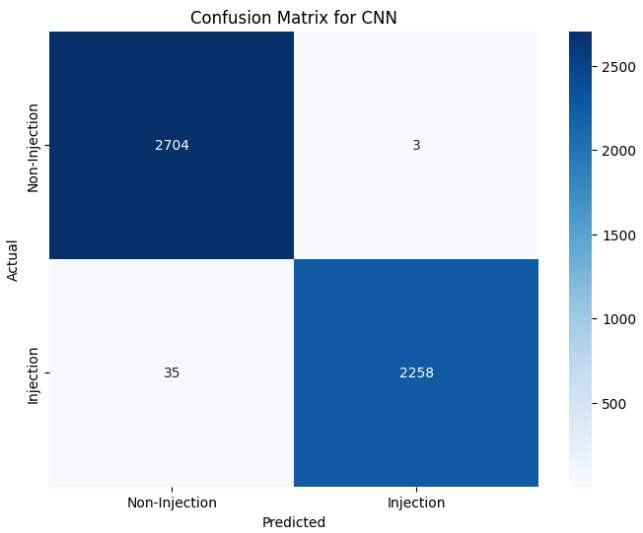



Confusion Matrix for CNN
Non-Injection
Injection
Actual
2704
3
35
2258
Non-Injection
Injection
Predicted
2500
2000
1500
1000
500