Міністерство освіти і науки України
Національний університет «Одеська політехніка»
Українсько-німецький навчально-науковий інститут
Кафедра кібербезпеки та програмного забезпечення

Палагін Олексій Олександрович
Студент групи НРЗ-181

КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

Метод виявлення та захисту від XSS атак за допомогою автоматичного моніторингу веб-сервера

Спеціальність:
125 Кібербезпека

Спеціалізація, освітня програма:
Кібербезпека

Керівник:
Соколов Артем Вікторович
к.т.н., доцент

Одеса – 2022

Міністерство освіти і науки України
Національний університет «Одеська політехніка»
Українсько-німецький навчально-науковий інститут
Кафедра кібербезпеки та програмного забезпечення

Рівень вищої освіти перший (бакалаврський)
Спеціальність 125 – Кібербезпека
Освітня програма – Кібербезпека

ЗАТВЕРДЖУЮ
Завідувач кафедри КБПЗ
_____
д.т.н., проф. А.А.Кобозєва
____ _____2022р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Палагіну Олексію Олександровичу

1. Тема роботи: *Метод виявлення та захисту від XSS атак за допомогою автоматичного моніторингу веб-сервера*
Керівник роботи: *Соколов А.В. к.т.н., доцент*.
затверджені наказом ректора від „17_” 05. 2022р. № 168-в.
2. Зміст роботи: *підвищення ефективності виявлення та відбивання XSS атак шляхом розробки та реалізація алгоритму виявлення та захисту від XSS атак за допомогою автоматичного моніторингу веб-сервера*.
3. Перелік графічного матеріалу*: блок-схема алгоритму, рисунки інтерфейсу програмного продукту.*

4. Консультанти розділів роботи

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв
Охорона праці	*доц. Ярова І.А.*		

Дата видачі завдання "_____"_____2022 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання	Примітка
1	*Аналіз літератури з теми кваліфікаційної роботи*	*20-03-2022*	*виконано*
2	*Аналіз існуючих методів виявлення XSS атак*	*28-03-2022*	*виконано*
3	*Аналіз системи захисту веб-додатків*	*05-04-2022*	*виконано*
3	*Модифікація згорткової нейронної мережі*	*20-04-2022*	*виконано*
4	*Розробка веб-додатку*	*29-04-2022*	*виконано*
5	*Розробка системи виявлення та захисту веб-додатка*	*10-05-2022*	*виконано*
6	*Підготовка пояснювальної записки.*	*20-05-2022*	*виконано*
7	*Підготовка презентації та доповіді*	*30-05-2022*	*виконано*
8	*Попередній захист*	*02-06-2022*	*виконано*
9	*Нормоконтроль, рецензування*	*18-06-2022*	*виконано*

Здобувач вищої освіти _____ *Палагін О.О.*

Керівник роботи _____ *Соколов А.В.*

ЗАВДАННЯ

на розробку розділу «Охорона праці» у кваліфікаційній роботі бакалавра

Палагіну Олексію Олександровичу, група НРЗ-181

Українсько-німецький навчально-науковий інститут

Кафедра кібербезпеки та програмного забезпечення

Дата отримання завдання 12.05.2022

Консультації 19.05.2022, 26.05.2022

Дата закінчення розділу 31.05.2022

Тема роботи: *Метод виявлення та захисту від XSS атак за допомогою автоматичного моніторингу веб-сервера*

Зміст розділу:

1. Аналіз умов праці і вибір основних заходів виробничої безпеки
2. Аналіз пожежної безпеки і вибір заходів і засобів пожежної безпеки

Керівник роботи	Консультант з охорони праці
_____ (А.В. Соколов)	_____ (Ярова І.А.)
«____» _____ 2022 р.	«____» _____2022 р

АНОТАЦІЯ

Кваліфікаційна робота на тему «Метод виявлення та захисту від XSS атак за допомогою автоматичного моніторингу веб-сервера» на здобуття першого (бакалаврського) рівня вищої освіти за спеціальністю 125 Кібербезпека, спеціалізація, освітня програма: Кібербезпека, містить 27 рисунків, 1 таблицю, 1 додаток, 26 літературних джерела за переліком посилань. Робота виконана на 60 сторінках загального тексту і 52 сторінках основного тексту.

Основною метою даної кваліфікаційнох роботи є розробка системи виявлення та захисту від XSS атак за допомогою автоматичного моніторингу веб-сервера завдяки згортковій нейронній мережі.

Розробка системи базується на аналізі систем захисту веб-додатків.

Програмно реалізовано згорткову нейрону мережу для класифікації запитів в веб-логах.

Дана система дозволяє підвищити рівень рівень точності виявлення XSS атак.

Результат роботи може бути використана для захисту веб-додатків.

ВЕБ-АТАКА, ВЕБ-ДОДАТОК, МАШИНЕ НАВЧАННЯ, СИСТЕМА ВИЯВЛЕННЯ XSS АТАКИ, ГЛИБОКЕ НАВЧАННЯ.

ANNOTATION

Qualification work on the topic "Method of detecting and protecting against XSS attacks using automatic web server monitoring" for the first (bachelor's) level of higher education in the speciality 125 Cybersecurity, specialization, educational program: Cybersecurity, contains 27 figures, 1 table, 1 appendix, 26 references links. The work is performed on 60 pages of general text and 52 pages of main text.

The main purpose of this qualification work is to develop a system for detecting and protecting against XSS attacks by automatically monitoring the web server through a convolutional neural network.

The development of the system is based on the analysis of web application protection systems.

A convolutional neural network has been programmatically implemented to classify queries in web logs.

This system allows you to increase the level of accuracy of detection of XSS attacks.

The result can be used to protect web applications.

WEB ATTACK, WEB APPLICATION, MACHINE LEARNING, XSS ATTACK DETECTION SYSTEM, DEEP LEARNING.

ЗМІСТ

ВСТУП

Динамічні веб-додатки відіграють важливу роль у забезпеченні маніпуляції ресурсами та взаємодії між клієнтами та серверами. Функції, які зараз підтримуються браузерами, розширили можливості для бізнесу, забезпечуючи високу інтерактивність у веб-сервісах, таких як веб-банкінг, електронна комерція, соціальні мережі, форуми, і в той же час ці функції створили серйозні ризики та збільшили вразливість у веб-додатки, які дозволяють здійснювати кібератаки.

Однією з поширених кібератак високого ризику на вразливості веб-додатків є міжсайтові сценарії (XSS). У наш час XSS все ще різко зростає і вважається однією з найсерйозніших загроз для організацій, користувачів і розробників.

Уразливості XSS можна легко використати, оскільки існує багато вільно доступних інструментів, які дозволяють кожному, хто має мінімальні знання, атакувати веб-додатки. Атаки XSS можуть призвести до викрадення сеансу, розкриття конфіденційних даних, атак підробки міжсайтових запитів (csrf) та інших вразливостей безпеки, включаючи видавання себе за жертву. Атаки XSS також можуть призвести до виконання коду на сервері, залежно від програми та привілеїв облікового запису користувача.

Щоб впоратися з веб-атаками, було досліджено та застосовано ряд методів захисту веб-додатків, веб-сайтів та користувачів Інтернету. Серед них виявлення веб-атак є перспективним підходом на захисних рівнях для захисту веб-сайтів і веб-додатків. Однак деякі методи вимагають регулярного оновлення правил виявлення або потребують великих обчислювальних ресурсів, оскільки вони використовують складні методи виявлення.

Метою даної кваліфікаційної роботи є підвищення ефективності виявлення та відбивання XSS атак шляхом розробки та реалізація алгоритму виявлення та захисту від XSS атак за допомогою автоматичного моніторингу веб-сервера.

Для досягнення даної мети були поставлені наступні задачі:

- створення сайта для проведення атаки;
- створення нейронної мережі;
- створення функцій виявлення та протидії.

Об’єкт дослідження — протидія веб-атакам.

Предмет дослідження — метод виявлення та захисту від XSS атаки.

Дипломний проект складається з вступу, чотирьох розділів, висновків, списку використаних джерел та додатків.

Перший розділ роботи присвячений загальним питанням та аналізу існуючих методів виявлення XSS атаки для веб-додатків.

У другому розділі запропонована система виявлення та захисту від XSS атак за допомогою автоматичного моніторингу веб-сервера.

Третій розділ є описом розробленої системи виявлення та її захисту від XSS атаки.

Четвертий розділ містить деякі із заходів, спрямованих на забезпечення вимог з охорони праці та екології.

1 АНАЛІЗ МЕТОДІВ ВИЯВЛЕННЯ XSS АТАКИ

1.1 XSS атака

Атака XSS — це підмножина атаки з ін’єкцією, коли зловмисник впроваджує шкідливий код у вміст законних і надійних веб-сайтів, щоб отримати контроль доступу до системи глядача. Атака XSS може бути виконана на будь-якому вразливому веб-сайті, будь то той, який написаний у коді HTML, JavaScript, VBScript, PHP тощо. Характер атаки залежить від типу атак XSS.

Наступні кроки пояснюють загальний сценарій атаки.

Крок 1. Зловмисник знаходить вразливий веб-сайт, який дозволяє вводити ненадійний шкідливий код на його веб-сторінку. Наприклад, розміщення фальшивої реклами на веб-сторінці, відображення неправдивого вмісту на веб-сайті.

Крок 2. Зловмисник вставляє шкідливий код JavaScript/ActiveX/VBScript/HTML на стороні клієнта у веб-додаток. Цей код надсилається або на веб-браузер жертви, або на веб-сервер залежно від типу атаки XSS.

Крок 3. Користувач натискає шкідливе посилання під час відвідування веб-сайту або доступу до служби з веб-сервера.

Крок 4. Зловмисник має доступ до приватних облікових даних або деталей жертви через вразливий веб-сайт, минаючи SOP [1].

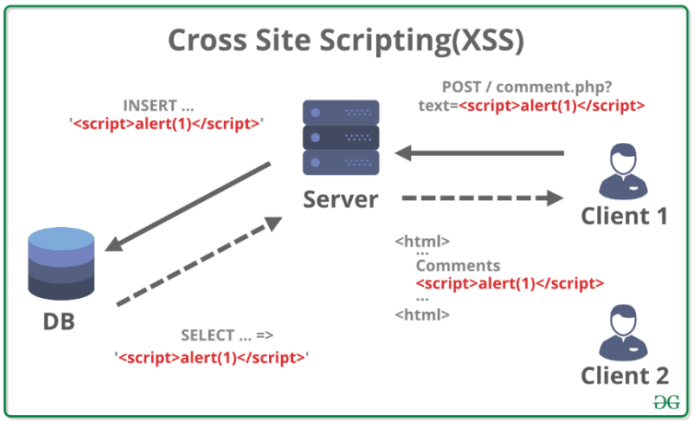

Рисунок 1.1 – XSS атака

1.1.1 Постійний/збережений XSS

Постійний XSS також відомий як збережений XSS. У цій атаці шкідливий сценарій додається безпосередньо на веб-сайт (особливо у форми, блоги або розділи коментарів), тому вона також відома як збережена XSS атака, оскільки сценарій зберігається на веб-сервер. Таким чином, щоразу, коли користувач відвідує цей веб-сайт, шкідливий код виконується, отже, вважається, що вона є більш шкідливою, ніж інші два типи, оскільки вона навіть не вимагає від користувача натискання посилання і може легко проникнути до кількох користувачів одночасно, тому визначити цей тип нападу досить складно. Наступний рисунок (рис. 1.2) представляє зразок сценарію збереженої XSS-атаки [1].

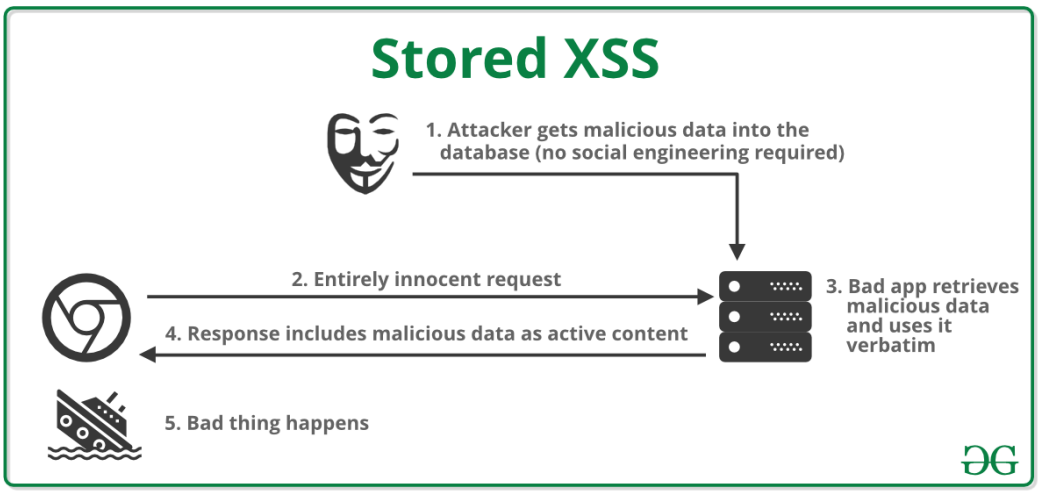

Рисунок 1.2 – Збережений тип атаки

1.1.2 Відображений/непостійний XSS

Непостійна атака XSS також називається відбитою XSS-атакою, де відбита означає, що результати шкідливого запиту є видимими для зловмисника. Зловмисник створює шкідливе посилання таким чином, щоб воно було з надійного джерела. Коли жертва натискає шкідливе посилання, веб-сервер надсилає користувачеві

відповідь, що містить шкідливий сценарій. На наступному рисунку (рис. 1.3) представлений приклад сценарію відбитої XSS-атаки [1].

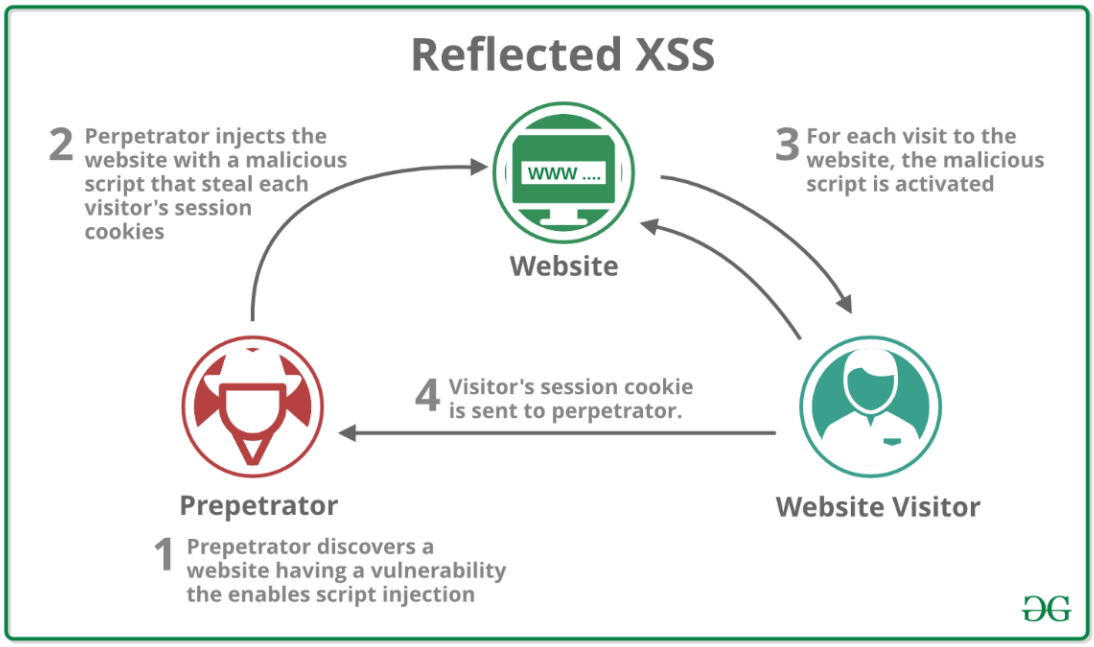

Рисунок 1.3 – Відображений тип атаки

1.1.3 XSS на основі DOM

Іншим значним вектором атаки є атака XSS на основі DOM, яка вплинула на найвідоміші організації світу, такі як Google, Yahoo та Amazon.

Хоча його ефект помірний, він все ще викликає серйозне занепокоєння для веб-додатків. XSS-атака на основі DOM – це атака на уразливість на стороні клієнта, коли зловмисник змінює вміст статичної або динамічної сторінки DOM на стороні клієнта за допомогою властивостей документа, таких як document.write, document.url або document.referrer . Отже, код на стороні клієнта виконується небажаним чином, і веб-користувач буде повністю не усвідомлювати атаку, оскільки вміст веб-сторінки, що відображається, не буде змінено.

DOM означає об'єктну модель документа. Це API, який використовується для доступу до вмісту або властивостей веб-сторінки, також відомої як документ.

Іншими словами, DOM є об’єктною моделлю для кожної веб-сторінки на основі HTML або XML. Отже, коли виконується XSS-атака на основі DOM, код JavaScript вбудовується в програму на стороні клієнта, таким чином, змінюючи вміст DOM, а також може змінювати значення властивостей об’єктів. Оскільки вектор атаки розміщується на сторінці відповіді, а шкідливий код виконується на комп’ютері жертви, алгоритм виявлення на стороні сервера не зможе виявити цей тип атаки. На рисунку 1.4 показано приклад сценарію XSS-атак на основі DOM.

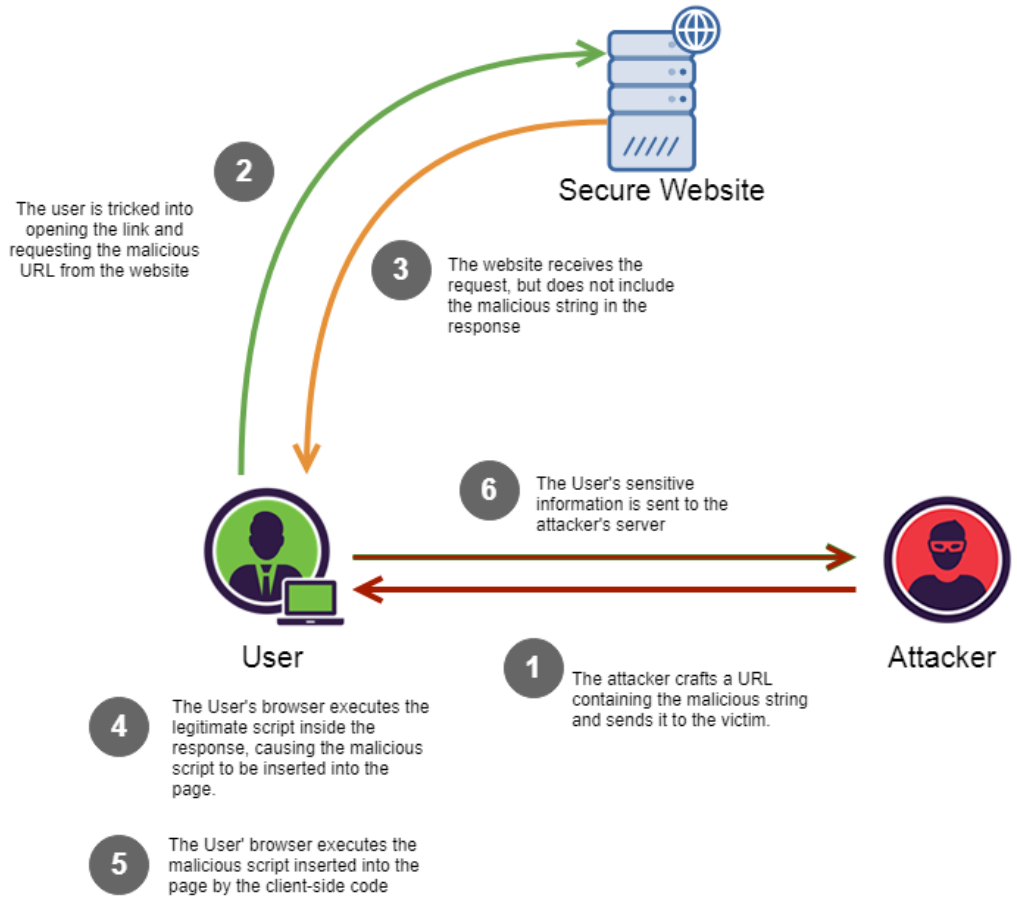

Рисунок 1.4 – XSS на основі DOM

1.2 Заходи боротьби з веб-уразливостями

Велика кількість дослідників запропонувало різні методології для боротьби з веб-уразливостями. На рисунку 1.5 представлені основні підходи безпеки веб-

додатків. Ці методи можна використовувати взаємно на різних етапах життєвого циклу розробки веб-додатків, і вони можуть бути реалізовані та розміщені на стороні клієнта або сервера веб-додатків.

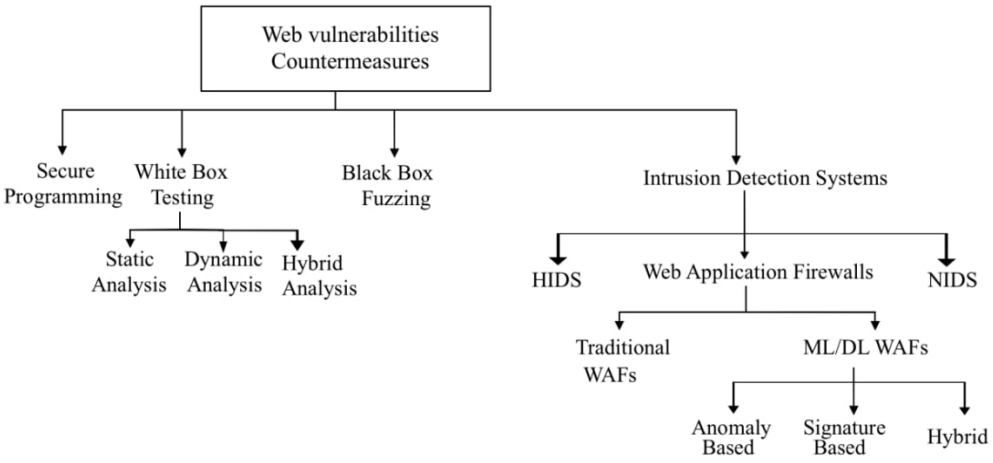

Рисунок 1.5 – Заходи боротьби з веб-уразливостями

1.2.1 Безпечне програмування

Оскільки атаки XSS реалізуються шляхом ін’єкції шкідливого сценарію на вразливу веб-сторінку, єдиний спосіб виявити незаконний сценарій із вихідного коду – це відфільтрувати вхідні дані та уникнути вихідних даних.

Фільтрація введених даних або перевірка введених даних здійснюється переважно на основі значень білого та чорного списку. У той час як перший гарантує, що введені дані повністю безпечні від будь-якого небажаного незаконного введення, вихідний екранування гарантує, що програма отримує захищені дані шляхом заміни escape-символів іншими символами, наприклад #, можна замінити на &#35, < на &it, > з &gt.

Проект OWASP (Open Web Application Security Project) запропонував різні стандарти, щоб дозволити розробникам дотримуватися безпечних практик під час кодування веб-додатків (ASVS (Application Security Verification Standard) [2], ESAPI (Enterprise Security API), SAMM (OWASP Software Assurance Maturity Model) [3]).

Незважаючи на те, що безпечне програмування може допомогти запобігти веб-уразливостям, воно забирає час, і його недостатньо через складність веб-додатків і різноманітність технологій і зовнішніх бібліотек, які беруть участь у розробці веб-додатків.

1.2.2 Тестування білої коробки

Під час тестування веб-програми як перевірки безпеки програмісти використовують статичний аналіз, динамічний або гібридний аналіз коду.

Статичний аналіз не вимагає виконання програми, і його можна виконати, просто перевіривши кожен рядок коду на наявність невідомої поведінки.

Існує кілька дослідницьких робіт щодо виявлення веб-уразливостей за допомогою статичного аналізу.

Наприклад:

- [4] запропонувала контекстний, міжфункціональний аналіз і аналіз потоків даних, щоб виявити вразливості в стилі плями, такі як SQL, ін’єкція команд та атаки XSS;
- [5] об’єднала аналіз вихідного коду (тобто відстежує введення користувачів, щоб перевірити, чи досягають вони чутливого приймача) ;
- [6] описала метод статичного аналізу, який автоматично виявляє вразливості контролю доступу у веб-додатках;
- [7] запропонувала метод статичного аналізу на основі машинного навчання для виявлення веб-вразливостей. Вони точно використали приховану марковську модель і анотовані фрагменти коду, щоб навчити модель виявляти вразливості у вихідному коді.

Загалом, інструменти на основі статичного аналізу є рішенням для пошуку вразливостей в Інтернеті, але вони, як правило, генерують хибні спрацьовування, вони займають багато часу і можуть ніколи не збігтися для великих баз коду.

Динамічний аналіз виконується шляхом аналізу програми з використанням реальних вхідних значень під час виконання програми. Тобто він запускає веб-додаток і намагається виявити порушення безпеки, використовуючи такі методи, як інструментування коду (вставлення перевірок у програму) і фаззінг (вводить випадкові тестові дані до цільової програми, щоб вивчити всі можливі шляхи).

Існує кілька дослідницьких робіт щодо виявлення веб-уразливостей за допомогою динамічного аналізу.

Наприклад:

- [8] покращила виявлення атак XSS у веб-додатках за допомогою динамічного аналізу та нечіткої системи. Вони витягли точки входу в додатки (AEP) за допомогою веб-сканера, а потім використали нечіткий механізм, який генерує недійсні рядки для кожного AEP, доки брандмауер веб-додатків не буде переможений, і в цьому випадку його база сигнатур оновлюється;
- [9] представила метод виявлення атак DOM-XSS: вони використовували динамічний аналіз забруднення коду JavaScript (тобто сліди плям отримують під час аналізу веб-сторінок) і фаззінг для автоматичного отримання векторів атаки на основі цих слідів плям, і потім вони перевіряють уразливість DOM-XSS, відтворюючи відповіді HTTP у браузері;
- [10] використовувала інструменти коду для створення моделей, які описують, як і з ким взаємодіють клієнтські компоненти, що дозволяє захистити веб-додатки на основі JavaScript від атак перевірки на стороні клієнта;

Підходи, засновані на динамічному аналізі, не викликають помилкових спрацьовувань, але не можуть досягти високого охоплення коду.

Гібридний аналіз передбачає поєднання методів статичного та динамічного аналізів. Однак ці методи тестування займають багато часу і не є ефективними для пошуку нових помилок у системі, які можуть змінити вміст веб-сторінки.

1.2.3 Тестування чорної коробки

Воно передбачає надсилання випадкових шкідливих даних до веб-програми, не враховуючи її логіку, і визначає, чи є програма вразливою, на основі її відповідей.

Фазери чорної скриньки зазвичай складаються з:

- сканер, який ідентифікує всі можливі веб-сторінки та точки входу в аналізованому веб-додатку;
- генератор тестових даних, який генерує випадкові дані для кожної точки входу програми;
- монітор, який виявляє помилки в поведінці програми під час виконання.

Існує кілька дослідницьких робіт щодо виявлення веб-уразливостей за допомогою тестування чорної коробки.

Наприклад:

- [11] запропонувала KameleonFuzz, чорний ящик XSS fuzzer для веб-додатків, який може генерувати шкідливі дані для використання вразливостей XSS, а також для виявлення того, наскільки близько він виявляє вразливість. Вони використовували генетичний алгоритм, керований граматикою атаки, для генерації та розвитку шкідливих даних;
- [12] запропонувала метод виявлення вразливості XSS, в якому використовували становий автомат для отримання знань про поведінку програми та генетичний алгоритм для автоматичного генерування вхідних даних із кращими значеннями пристосованості для ініціювання екземпляра даної вразливості.

Інструменти сканування веб-уразливостей також можна віднести до категорії «чорних скриньок». Вони сканують веб-додатки ззовні, щоб знайти вразливі місця в безпеці.

Їх часто використовують аналітики безпеки та хакери для пошуку вразливостей веб-додатків. Доступна велика кількість як комерційних, так і відкритих інструментів цього типу (Burp Suite, Nessus, Nekto тощо).

Фаззінг «чорного ящика» не вимагає наявності вихідного коду програми, але може призвести до високого рівня помилкових негативів і помилкових спрацьовувань залежно від ефективності сканера та генератора даних атаки, відповідно.

1.2.4 Системи виявлення вторгнень

Вони визначаються як системи, створені для моніторингу хост-систем (система виявлення вторгнень (HIDS)) або мережевих комунікацій (система виявлення вторгнень в мережі (NIDS)) або веб-додатків (брандмауер веб-додатків (WAF)).

HIDS зазвичай використовуються для виявлення шкідливих програм.

NIDS зазвичай виявляють мережеві атаки, такі як DoS (відмова в обслуговуванні), атаки MITM (Man-In-the-Middle), а також деякі типи веб-атак.

WAF допомагають захистити веб-програми, фільтруючи та відстежуючи HTTP-трафік між веб-додатком та Інтернетом. Вони виявляють веб-атаки на стороні клієнта (наприклад, атака DOM-XSS) і на стороні сервера (наприклад, атака SQL-ін’єкції).

Розрізняються традиційні WAF і WAF на основі машинного навчання (ML) [13] або алгоритмів глибокого навчання (рис 1.6).

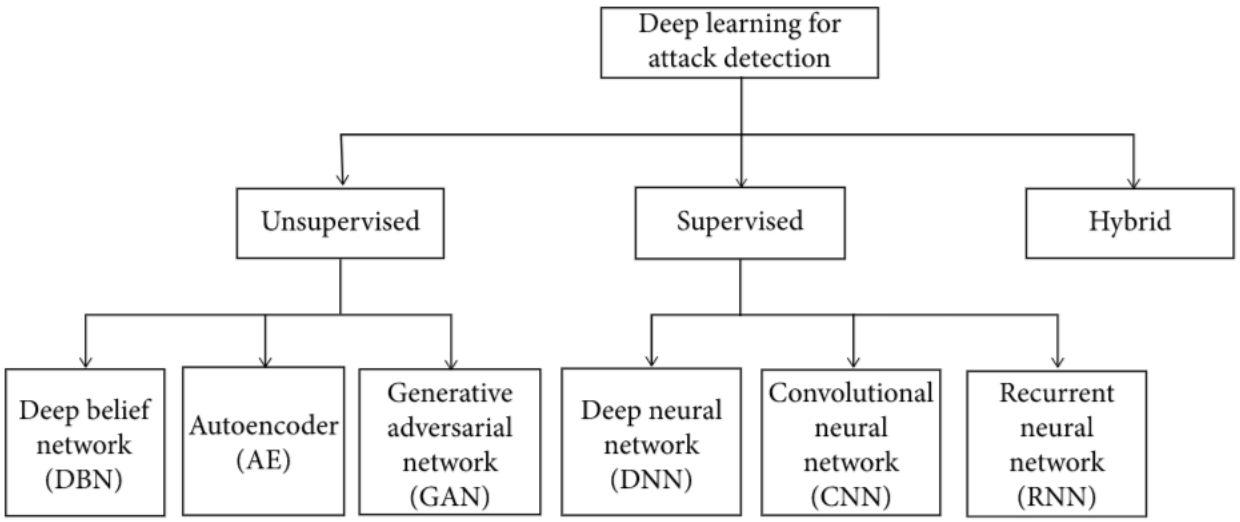

Рисунок 1.6 - Моделі глибокого навчання, які можуть бути використані для виявлення веб-атак

Традиційні WAF використовують статичні правила зіставлення шаблонів для виявлення атак. Таким чином, вони не можуть виявити нові-невідомі атаки. Крім того, оновлення правил може бути виснажливим завданням, якщо схема атаки є складною. Однак вони генерують кілька помилкових спрацьовувань.

Що стосується WAF на основі ML/DL, то існує три основні підходи:

- підхід на основі аномалій, за яким моделі навчаються з використанням звичайних екземплярів даних унікальних і неконтрольованих або гібридних алгоритмів ML;

- підхід на основі сигнатур (найбільш використовується), в якому моделі навчаються із звичайними та аномальними екземплярами даних та автономними контрольованими алгоритмами ML;

- гібридний підхід, який поєднує підхід на основі сигнатур і підхід виявлення аномалій.

Перший підхід може виявити атаки нульового дня, але страждає від високої частоти помилкових спрацьовувань, оскільки неочевидно визначити з упевненістю, що є нормальним у певній області.

Другий тип підходу не може виявити атаки нульового дня, але може виявляти відомі атаки точно та з меншою кількістю помилкових спрацьовувань, ніж перший підхід.

Третій підхід може зробити найкраще з обох підходів. Він може досягти високої точності при виявленні відомих атак, одночасно генеруючи низьку кількість помилкових спрацьовувань при виявленні нових невідомих атак. Крім того, базу даних сигнатур IDS можна покращити, додавши сигнатуру щойно виявлених атак.

Проаналізувавши системи виявлення веб-атак за допомогою DL, можна сказати, що найбільш популярні CNN, LSTM та DFFN.

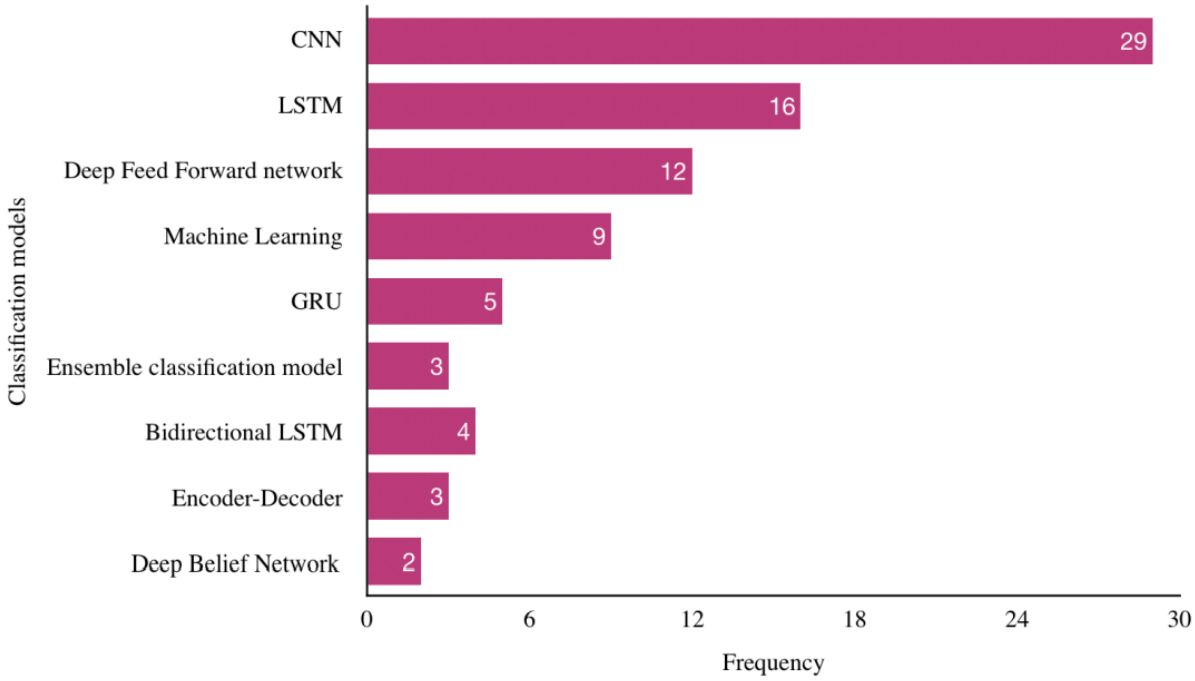

Рисунок 1.7 - Гістограма використаних моделей класифікації

Незважаючи на це, CNN моделей для виявлення XSS атаки за допомогою перевірки веб-логів сервера не було знайдено. Найбільш близька до цього робота використовувала традиційні методи машинного навчання [14]. Тому перейдемо до розробки своєї моделі виявлення XSS атаки.

2. РОЗРОБКА СИСТЕМИ ВИЯВЛЕННЯ ТА ЗАХИСТУ ВІД XSS АТАК ЗА ДОПОМОГОЮ АВТОМАТИЧНОГО МОНІТОРИНГУ ВЕБ-СЕРВЕРА

2.1 Опис системи

Створена система виявлення веб-атак складається із двох етапів:

- навчання;
- виявлення та протидії.

Етап навчання складається із 3 кроків.

Крок 1. Збирання екземплярів XSS атаки, та звичайних комментарів користувачів із різних баз даних для формування одного великого датафрейму.

Крок 2. Обробка даних цього датафрейму завдяки використовуваню кодування Unicode кожного символу.

Крок 3. Масив оброблених даних поміщається в крок «Навчання» для побудови класифікатора.

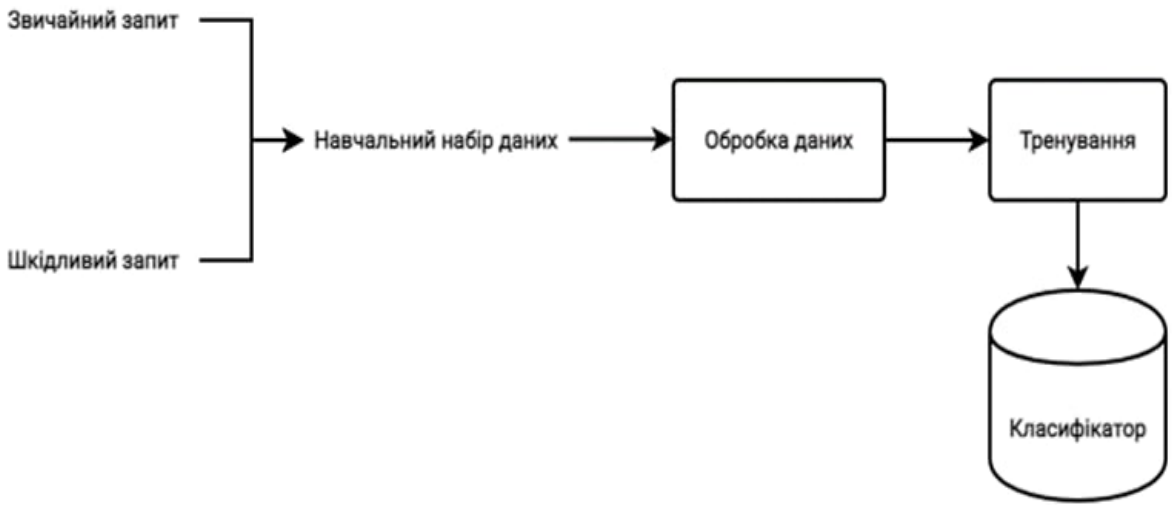

Рисунок 2.1 – Етап навчання

Етап виявлення та протидії складається із 4 кроків.

Крок 1. Вилучення запитів із веб-логів сервера завдяки регулярним виразам.

Крок 2. Обробка запиту таким же чином, як і дані тренувального датафрейму.

Крок 3. Створена модель класифікації (класифікатор) отримує вхідними даними оброботаний масив, після чого видає на вихід мітку «0» або «1», де «0» - звичайний лог, а «1» - веб-атака.

Крок 4. Додування ір користувача до чорного списку, у разі виявлення XSS атаки.

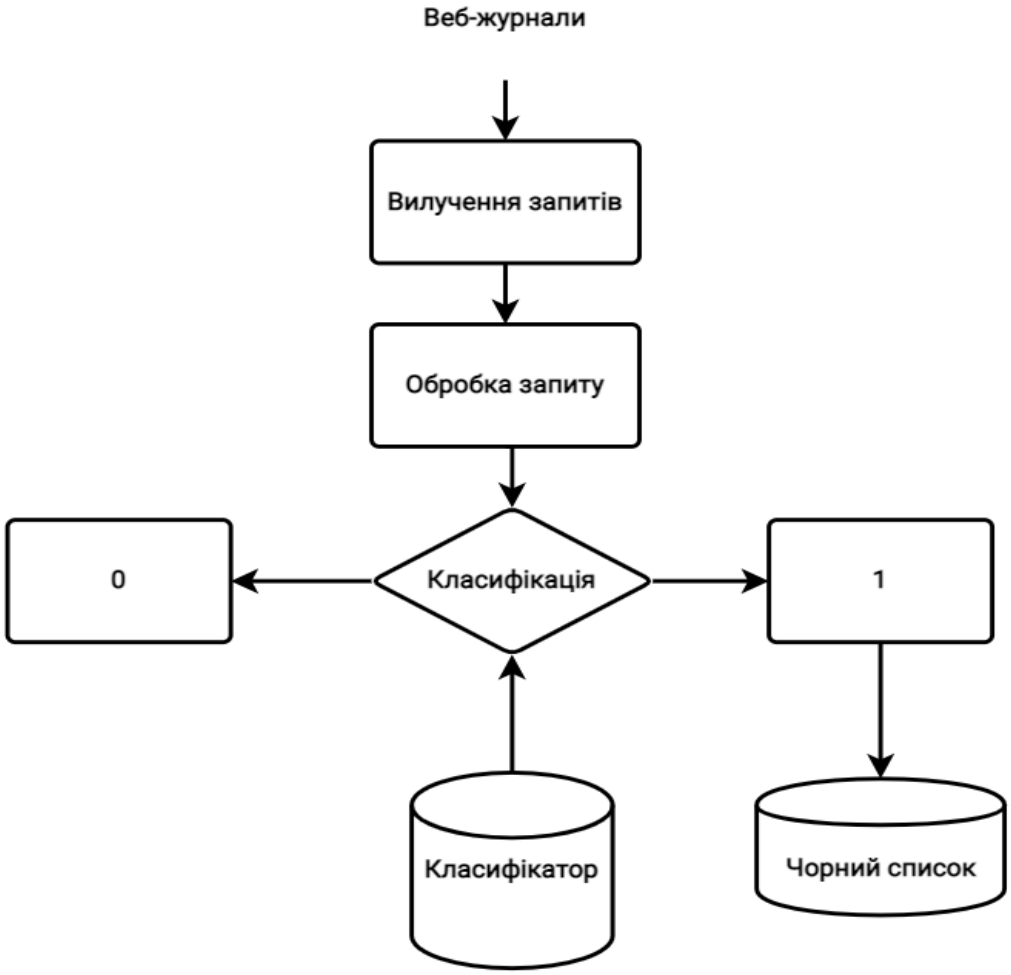

Рисунок 2.2 – Етап виявлення та протидії

2.2 Обробка даних

В цій системі використовується кодування Unicode кожного символу (число в таблиці Unicode, яке представляє символ). Оскільки нейронна мережа насправді не розуміє слова, скоріше це функція, котра сприймає математичні об’єкти, такі як

числа, вектори та матриці, тому дані перед тим як потрапляти до мережи проходять через функцію обробки.

```
<isindex draggable="true" ondragleave="alert(1)">test</isindex>

[ 60, 105, 115, 105, 110, 100, 101, 120,  32, 100, 114,  97, 103, 103,  97,  98, 108, 101,  61,  34, 116, 114, 117, 101,  34,  32, 111, 110],
[100, 114,  97, 103, 101, 110, 116, 101, 114,  61,  34,  97, 108, 101, 114, 116,  40,  49,  41,  34,  62, 116, 101, 115, 116,  60, 47, 105],
[115, 105, 110, 100, 101, 120,  62,  10,   0,   0,   0,   0,   0, 0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0, 0,   0]
```

Рисунок 2.3 – Запит і його Unicode формат

Це набагато кращий варіант у порівнянні з іншими типами кодування, такими як character embedding, word embedding, та one hot encoding.

Причина того, що він краще, полягає в тому, що він дуже ефективний у зберіганні. Звичайне character embedding зазвичай кодує символ у якийсь вектор, що вже є більш дорогим, ніж просто вбудовування символу в його числове представлення Unicode.

Далі закодовані запити будуть відформатовані в матриці розміром 28x28.

2.3 Опис згорткової нейронної мережи

CNN — це клас нейронних мереж, який спеціалізується на обробці даних, які мають топологію, подібну до сітки, наприклад зображення.

CNN зазвичай має три шари: згортковий шар, шар об’єднання та повністю підключений шар.

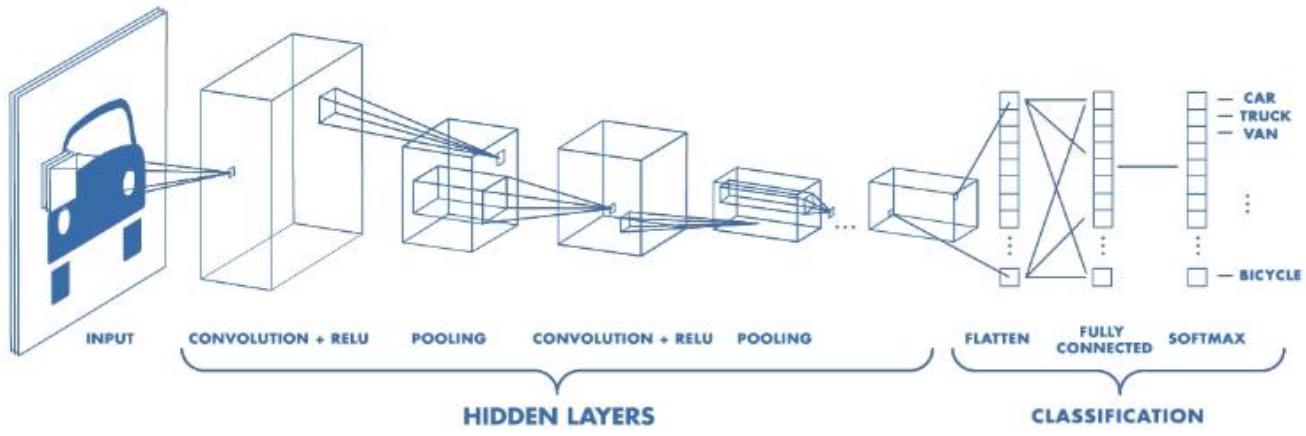

Рисунок 2.4 – Архітектура CNN

2.3.1 Згортковий шар

Шар згортки є основним будівельним блоком CNN. Він несе основну частину обчислювального навантаження мережі.

Цей рівень виконує точковий добуток між двома матрицями, де одна матриця є набір параметрів, які можна вивчати, інакше відомих як ядро, а інша матриця є обмеженою частиною сприйнятливого поля. Ядро просторово менше, ніж зображення, але є більш глибоким. Це означає, що якщо зображення складається з трьох каналів (RGB), висота і ширина ядра будуть просторово малими, але глибина поширюється на всі три канали.

Під час прямого проходу ядро «ковзає» по висоті та ширині зображення, створюючи зображення цієї сприйнятливої області. Це створює двовимірне представлення зображення, відоме як карта активації, яка дає відповідь ядра в кожному просторовому положенні зображення. Ковзний розмір ядра називається кроком [15].

Згортання використовує три важливі ідеї, які мотивували дослідників комп’ютерного зору: розріджена взаємодія, спільне використання параметрів і еквіваріантне представлення.

Тривіальні шари нейронної мережі використовують множення матриці на матрицю параметрів, що описують взаємодію між вхідним і вихідним блоком. Це означає, що кожен вихідний блок взаємодіє з кожним вхідним блоком. Однак нейронні мережі згортки мають розріджену взаємодію. Це досягається шляхом зменшення розміру ядра, ніж вхідний, наприклад, зображення може мати мільйони або тисячі пікселів, але під час обробки його за допомогою ядра ми можемо виявити значущу інформацію, яка складається з десятків або сотень пікселів. Це означає, що нам потрібно зберігати менше параметрів, що не тільки зменшує потребу моделі в пам’яті, але й покращує статистичну ефективність моделі.

Завдяки спільному використанню параметрів, шари нейронної мережі згортки матимуть властивість еквіваріантності трансляції. У ньому сказано, що якщо вхідні дані були змінені, вихідні дані також будуть змінені таким же чином.

2.3.2 Шар об’єднання

Рівень об’єднання замінює вихід мережі в певних місцях шляхом отримання підсумкової статистики найближчих вихідних даних. Це допомагає зменшити просторовий розмір представлення, що зменшує необхідну кількість обчислень і значень. Операція об’єднання в пул обробляється для кожного фрагмента представлення окремо.

Існує кілька функцій об’єднання, наприклад середнє значення прямокутного околиці, норма L2 для прямокутного околиці та середньозважене значення на основі відстані від центрального пікселя. Однак найпопулярнішим процесом є максимальний пул, який повідомляє про максимальний вихід із сусідства [15].

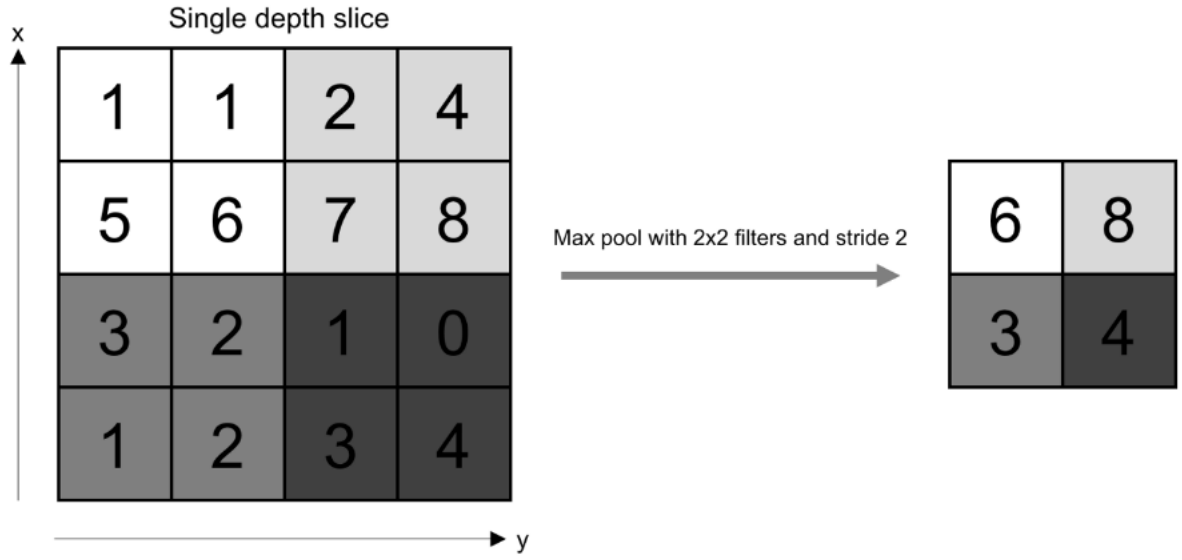

Рисунок 2.5 – Операція об'єднання

У всіх випадках об’єднання в пул забезпечує деяку інваріантність перекладу, що означає, що об’єкт буде впізнаним незалежно від того, де він з’являється у кадрі.

2.3.3 Повністю підключений шар

Повністю підключений шар (також відомий як прихований шар) є останнім шаром згорткової нейронної мережі. Цей шар є комбінацією афінної та нелінійної функції.

Афінна функція: y = Wx + b

Нелінійна функція: Sigmoid, TanH і ReLu

Повністю підключений шар отримує вхідні дані від Flatten Layer, який є одновимірним шаром (1D Layer). Дані, що надходять від Flatten Layer, спочатку передаються до афінної функції, а потім до нелінійної функції. Комбінація 1 афінної функції та 1 нелінійної функції називається 1 FC (повністю підключений) або 1 прихований шар [16].

Ми можемо додати кілька таких шарів на основі глибини, на яку ми хочемо прийняти нашу модель класифікації. Зауважте, що це повністю залежить від набору навчальних даних. Вихідні дані з кінцевого прихованого шару надсилаються до функції Softmax або Sigmoid для розподілу ймовірності на кінцевий набір загальної кількості класів.

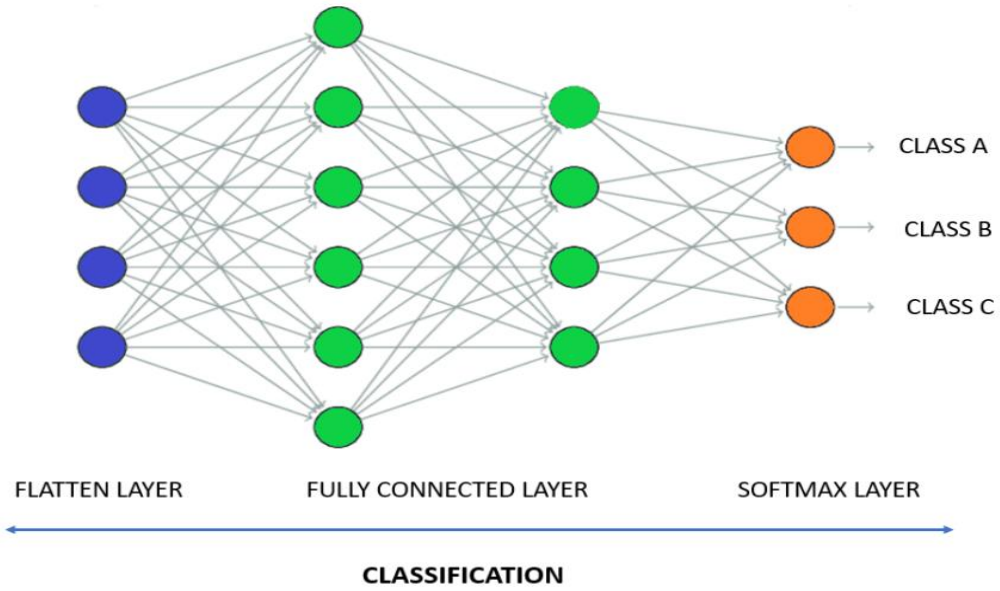

Рисунок 2.6 – Модель класифікації

Якщо подивитися на повну нейронну мережу, то побачимо, що початкові шари згорткової нейронної мережі складаються з:

- згортковий шар;
- об’єднаний шар;
- випадаючий шар.

Ці три разом охоплюють вибір (вилучення) функцій. На основі даних навчання можна додати різні перестановки та комбінації цих шарів.

Вихідний шар у згортковій нейронній мережі складається з:

- softmax або Sigmoid;
- розрахунок втрат за допомогою функції крос-ентропії.

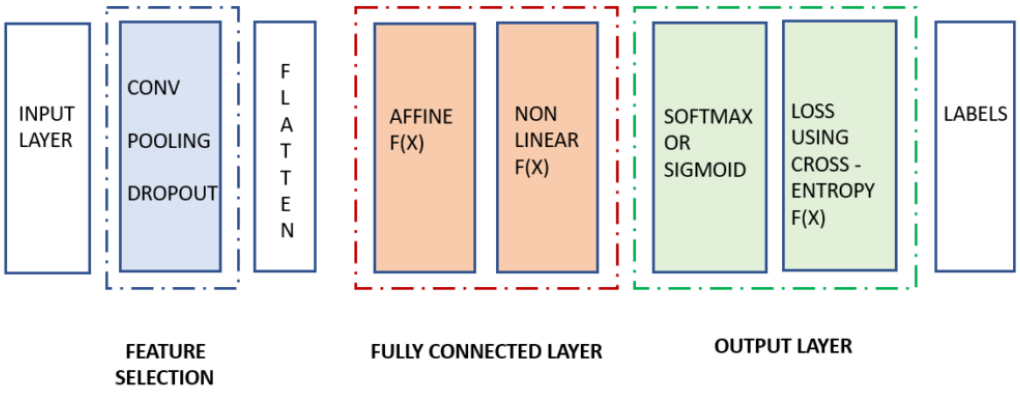

Рисунок 2.7 – Архітектура CNN

2.4 Опис створеної мережі

За основу була взята згорткова нейронна мережа для класифікації набору даних MNIST [17].

Перелік шарів мережи:

- шар згортки (convolution layer), де ядро шару матриця 5 на 5, а її розмірність 64;

- шар пакетного навчання нормалізації (BatchNormalization), використовується як регуляризація, щоб уникнути переобладнання моделі. Шар додається до послідовної моделі, щоб стандартизувати вхідні або вихідні дані;
- шар згортки (convolution layer), де ядро шару матриця 5 на 5, а її розмірність 32;
- випадаючий шар (dropout layer), маска, яка зводить нанівець внесок деяких нейронів у наступний шар і залишає незмінними всі інші;
- шар об’єднання (max pooling), де ядро витягує максимальне значення площі, яку воно згортає;
- згладжування (flatten), включає в себе взяття об’єднаної карти об’єктів, яка створюється на шарі об’єднання, і перетворення її в одновимірний вектор;
- повністю підключений шар (dense layer), із 100 нейронів, з функцією активації relu;
- другий повністю підключений шар (dense layer), де вже тільки 2 нейрони, із функцією softmax, що дозваляє отримати на виході процентне співвідношення між першою та другою відповіддю.

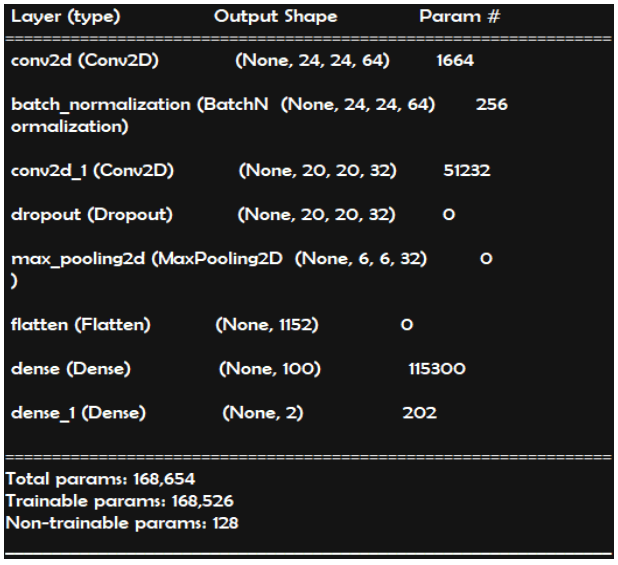

Рисунок 2.8 – Архітектура створеної мережі

Серед модифікацій було проведено:

- додавання ще одно згорткового шару (convolutional layer);

- додавання випадаючого шару (dropout) з коефіцієнтом 0.2;

- додавання ще 2 показників продуктивності, які зазвичай використовуються для оцінки моделей виявлення веб-атак на основі DL.

Отже, була розроблена система виявлення веб-атак, якщо всі компоненти будуть реалізовані, то XSS атака буде виявлена у разі сканування веб-логів.

3. РОЗРОБКА СИСТЕМИ ВИЯВЛЕННЯ XSS АТАК

3.1 Вибір середовища програмування

PyCharm — це спеціальне інтегроване середовище розробки Python (IDE), що надає широкий спектр важливих інструментів для розробників Python, тісно інтегрованих для створення зручного середовища для продуктивної розробки Python, вебу та науки про дані [18].

Коли PyCharm запускаєте вперше або коли немає відкритих проектів, можно побачити екран привітання. Він дає основні точки входу в IDE: створення або відкриття проекту, перевірка проекту з контролю версій, перегляд документації та налаштування IDE.

На рисунку 3.1 представлена графічна оболонка PyCharm.

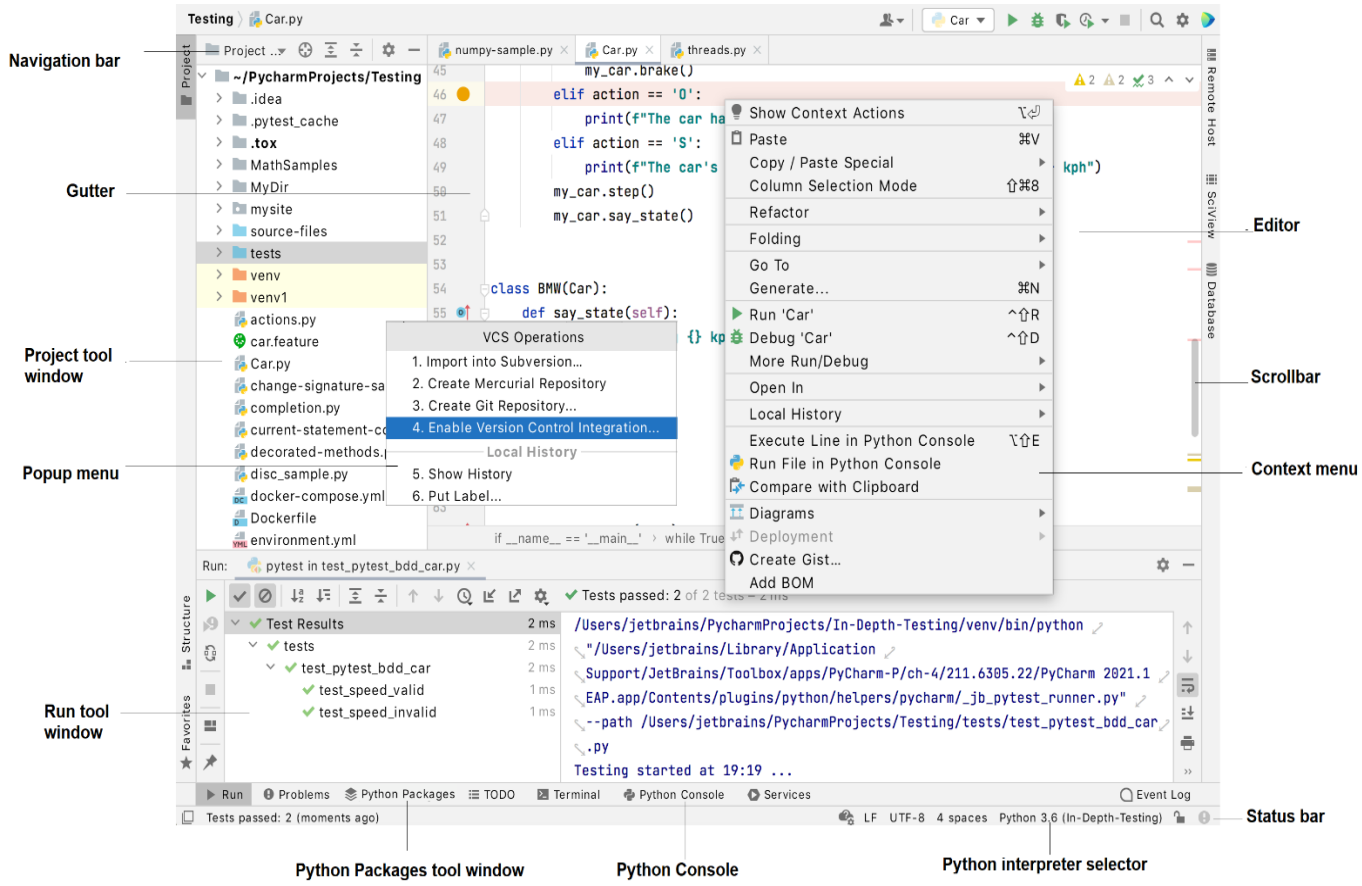

Рисунок 3.1 – Інтерфейс PyCharm

Коли проект відкривається, видно головне вікно, розділене на кілька логічних областей.

Ключові елементи інтерфейсу користувача:

- у вікні інструментів проекту (profect tool window) зліва відображаються файли проекту;

- редактор (editor) з правого боку, де пишуть код. Він має вкладки для легкої навігації між відкритими файлами;

- панель навігації (navigation bar) над редактором додатково дає змогу швидко запускати та налагоджувати програму, а також виконувати основні дії VCS;

- жолоб (gutter), вертикальна смуга поруч із редактором, показує наявні точки зупинки та забезпечує зручний спосіб навігації по ієрархії коду, як-от перехід до визначення/оголошення. Він також показує номери рядків та історію VCS для кожного рядка;

- смуга прокрутки (scroll bar), на правій стороні редактора. PyCharm постійно контролює якість вашого коду і завжди показує результати перевірок свого коду в стоку: помилки, попередження тощо. Індикатор у верхньому правому куті показує загальний стан перевірок коду для всього файлу;

- вікна інструментів (tool windows) — це спеціалізовані вікна, прикріплені знизу та з боків робочого простору і забезпечують доступ до типових завдань, таких як керування проектами, пошук вихідного коду та навігація, інтеграція з системами контролю версій тощо;

- рядок стану (status bar) вказує стан вашого проекту та всієї IDE, а також показує різні попередження та інформаційні повідомлення, як-от кодування файлів, роздільник рядків, профіль перевірки тощо. Він також забезпечує швидкий доступ до налаштувань інтерпретатора Python.

Оскільки створення веб-додатка планувалося на Django, веб-фреймворк Python високого рівня, а нейронну мережу на TensorFlow, бібліотеці програмного забезпечення машинного навчання, то питань відносно вибору середовища не було.

3.2 Робота із веб-додатком

Для того щоб продемонструвати XSS атаку, треба мати ціль, котру ви збираєтесь атакувати. Але, оскільки за подібну діяльність можливо отримати штраф, або навіть кримінальну відповідальність за статею 361 кримінального кодексу України [19], то у якості жертви для проведення атаки було обрано створення власного веб-додатку.

Для його створення був обраний Django, веб-фреймворк Python високого рівня.

На цьому веб-додатку була створена спеціальна сторінка, котра має одну уразливість – неекранований текст користувачів.

Рисунок 3.2 – Уразлива сторінка

На даній сторінці користувачі мають право залишати коментарі, що дає нам можливість для проведення stored XSS та reflected XSS атак.

Рисунок 3.3 – Введення XSS атаки

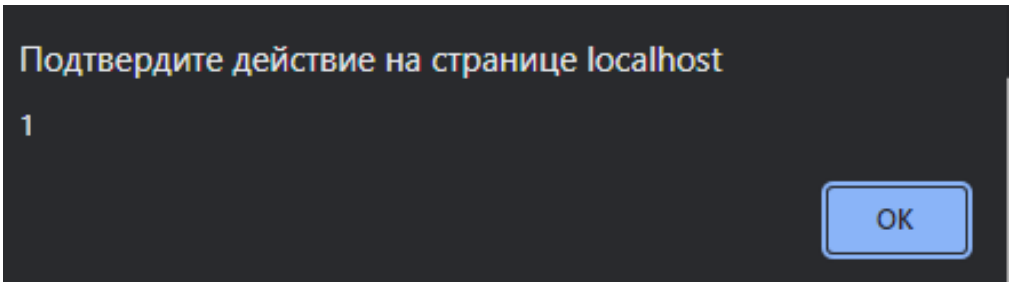

Рисунок 3.4 – XSS атака

Оскільки введена XSS атака відноситься до збереженого типу (stored), то після залишення такого коментаря, усі хто будуть відкривати цю сторінку, будуть отримувати цю атаку від серверу, де вона і зберігається.

3.3 Робота з програмами

Було створено 2 програми, одна для тренування моделі та перевірки її ефективності, інша для перевірки веб-логів на наявність шкідливих логів, та додавання відправників цих логів до чорного списку.

3.3.1 Програма тренування та тестування

Програма тренування та тестування була розроблена заради створення та збереження моделей нейронної мережі, та у їх подальшій перевірці на ефективність, а також наявність помилок першого та другого роду.

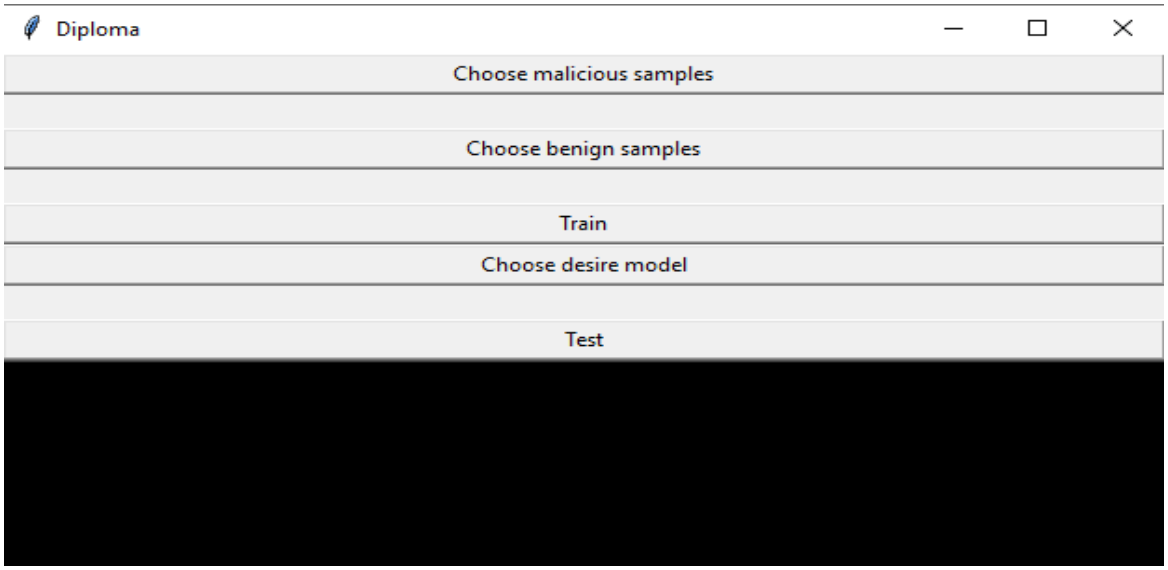

Рисунок 3.5 – Інтерфейс програми тренування та тестування

На рисунку 3.5 можна побачити що вона має спеціальні кнопки для вказання шляху до тренувальних екземплярів шкідливих та звичайних запитів. Також є кнопки для старту тренування та тестування.

У разі тренування, потрібно мати 2 обраних шляхи для обох видів запитів, після чого вони будуть об’єднані у один великий датафрейм, та оброблені функцією обробки для проходження тренування мережі.

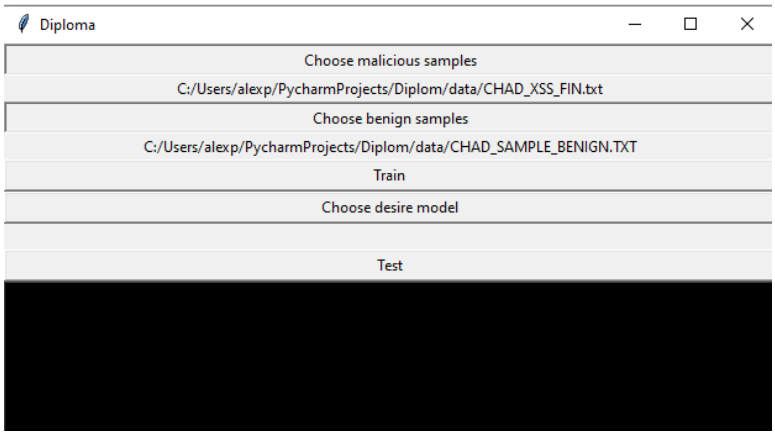

Рисунок 3.6 – Підготовка до тренування

Результатом виконання функції тренування програми із параметрами на рис 3.6 буде створені графіки із метрикою точність (accuracy), показниками функції втрат (loss function), та створення confusion matrix, тобто матриці із помилками першого та другого роду.

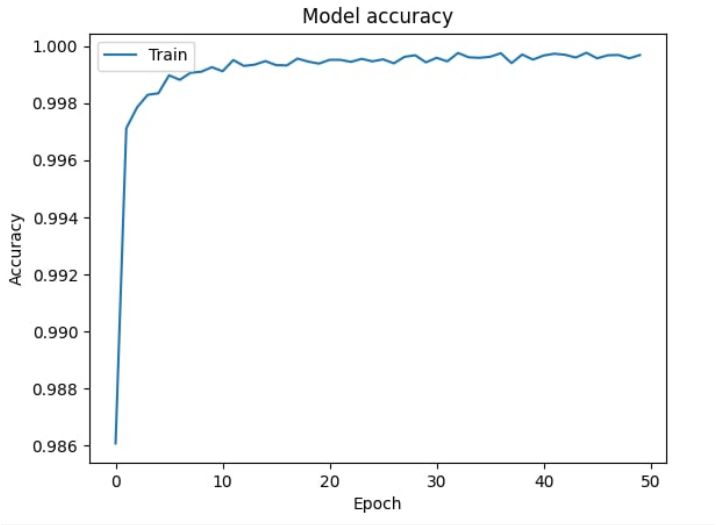

Рисунок 3.7 – Показники метрики точність

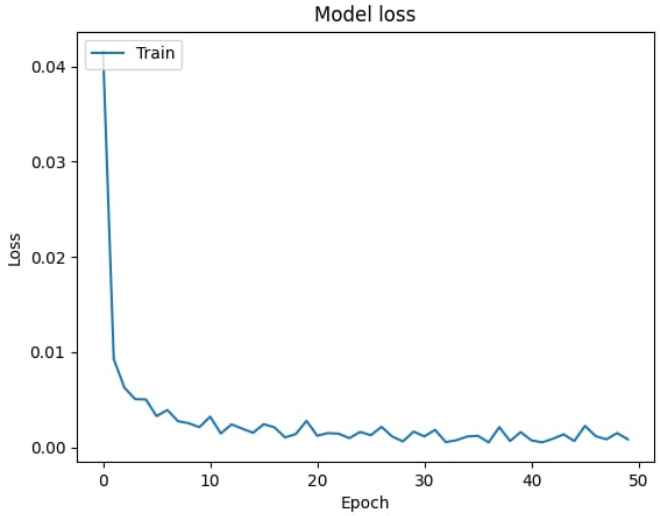

Рисунок 3.8 – Показники функції втрати

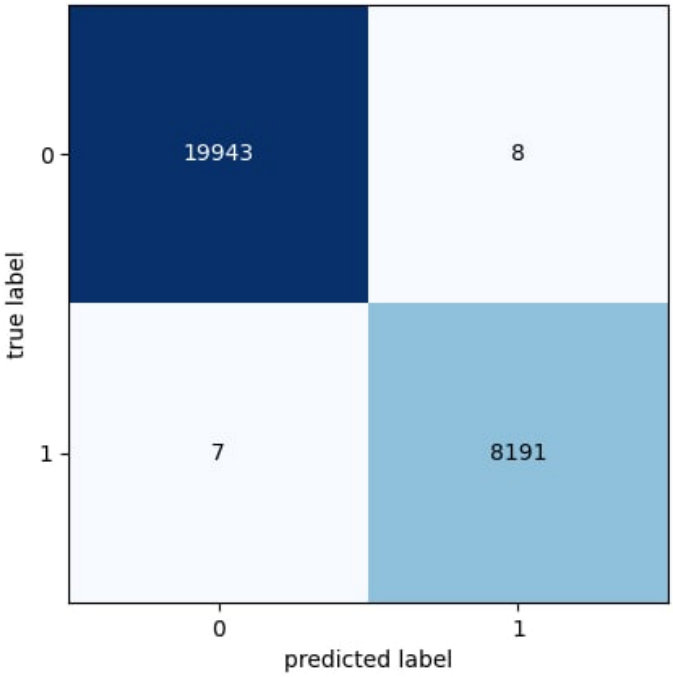

Рисунок 3.9 – Матриця помилок першого та другого роду

Де помилкою першого роду є лівий нижній квадрат, із значенням 7, а другого роду є правий верхній, із значенням 8.

Завдяки цим трьом результатам ми можемо оцінити із яким темпом навчалась мережа, та її ефективність.

Також у системи є функція тренування, для неї потрібно лише обрати один файл із запитами, оскілька програма не отримує познаки із міткою, до якого типу відноситься запити у файлі, до того ж обоє виду знаходяться у файлі. Після обирання файлу та зупску функії тренування, програма відповість, скільки вона знайшла шкідливих та звичайних запитів.

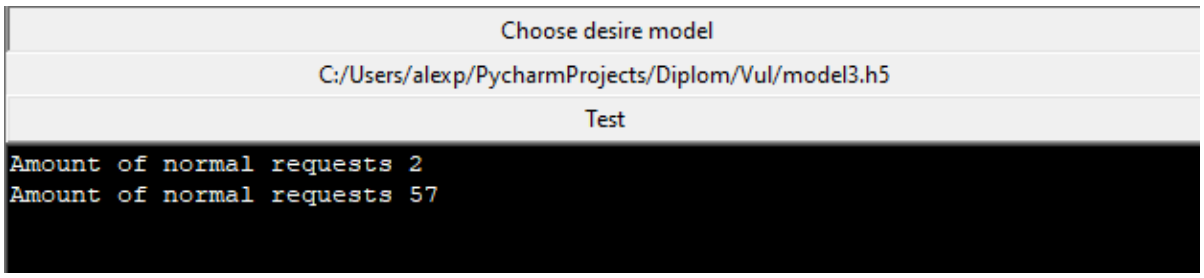

Рисунок 3.10 – Результат тестування програми

3.3.2 Програма виявлення та протидії XSS атаки

Інтферфейс програми досить мінімалістичний, оскільки модель тренування та файл веб-журналу вшиті в програмномуу забезпечені програми. Тому там розташовані лише кнопка для запуску функції, та вікно для тексту, куди програма подає результати виконання.

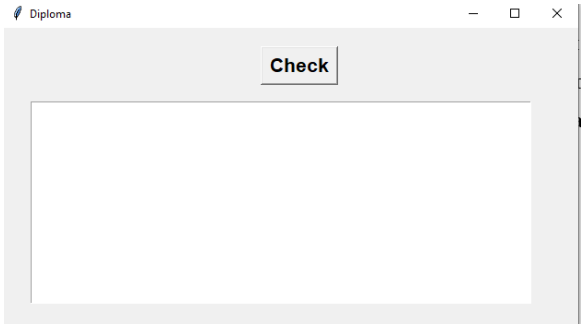

Рисунок 3.11 – Інтерфейс програми

В момент виконання функції, програма загружає модель класифікатора, отримує запити разом із ір коричтувачів з файлу веб-логів, оброблює запити, та надсилає їх до моделі. У разі виявлення XSS атаки, програма надсилає ір автора запиту до чорного списку.

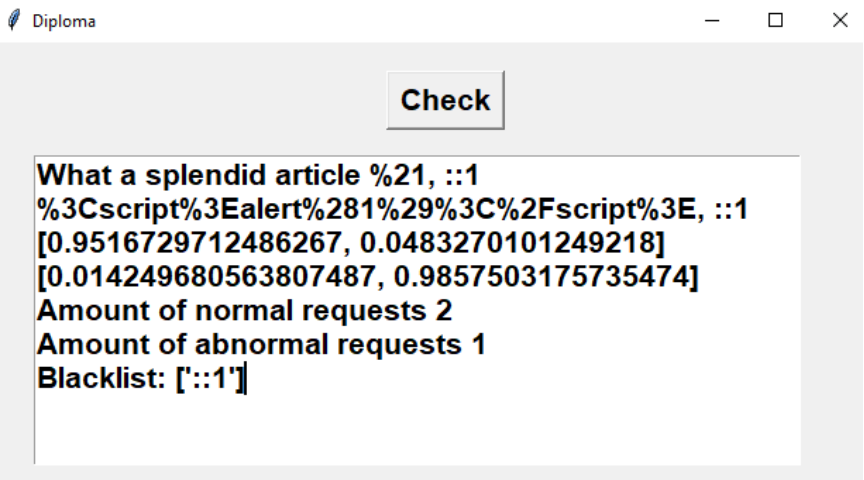

Рисунок 3.12 – Результат виконання функції

В розділі розглянуто опис розробленої системи виявлення та її захисту від XSS атаки.

4 ОХОРОНА ПРАЦІ

Охорона праці - система правових, соціально-економічних, організаційно-технічних, санітарно-гігієнічних і лікувально-профілактичних заходів та засобів. Мета котрих є спрямованість на збереження життя, здоров'я і працездатності людини у процесі трудової діяльності [20].

В даному розділі охорони праці були виявлені основні небезпечні і шкідливі виробничі фактори при роботі в офісному приміщені, та приведені групи заходів захисту, щодо зменшення впливу цих самих факторів за роботою персонального комп’ютера.

Перелік групи заходів захисту:

- заходи забезпечення санітарно-гігієнічних норм;
- заходи нормалізації параметрів виробничого мікроклімату;
- заходи забезпечення робочого освітлення;
- заходи захисту від шуму;
- заходи електробезпеки;
- заходи захисту від електромагнітного випромінювання.

Оскільки тема кваліфікаційної роботи є розробка та реалізація алгоритму виявлення та захисту від XSS атак за допомогою автоматичного моніторингу веб-серверу, то робочим місцем для спеціаліста з інформаційної безпеки для вирішення цієї задачі потрібен стіл, робоче крісло, та розміщений на ньому персональний комп’ютер, або ноутбук, а приміщення в якому усе це знаходиться буде офіс.

Відповідно до вимог роботи з персональним комп’ютером, або ноутбуком, робочі місця повинні відповідати безпечним умовам праці.

Безпечні умови праці - стан умов праці, за якого вплив на працівника небезпечних і шкідливих виробничих чинників усунуто або вплив шкідливих виробничих чинників не перевищує граничнодопустимих значень [21].

Підприємства повинні забезпечити ці умови праці, тому що їх недотримання може призвести до захворюваності працівників. При цьому необхідно пам’ятати, що будь-яке устаткування або технологічний процес завжди є джерелом певної кількості небезпечних і шкідливих виробничих факторів.

Шкідливі виробничі фактори - фактори середовища або трудового процесу, вплив яких на працівника за певних умов (інтенсивність, тривалість дії тощо) може спричинити професійне або виробниче обумовлене захворювання, тимчасове або стійке зниження працездатності, підвищення частоти соматичних та інфекційних захворювань, призвести до порушення здоров’я працівника [22].

Санітарно-гігієнічні вимоги до умов праці в офісних приміщеннях:

- площа на одне робоче місце має становити не менше ніж 6,0 кв. м.;
- об'єм на одне робоче місце має становити не менше ніж 20,0 куб. м.;
- відстані між бічними поверхнями ПК не менше ніж 1,2 м.;
- відстані від тильної поверхні одного ПК та екрана від іншого - 2,5 м.;
- висота робочої поверхні робочого столу з ВДТ має регулюватися в межах 0,68 - 0,8 м.;
- робочі місця повинні бути розташовані на відстані не менше ніж 1 м від стіни з вікном.

Під мікрокліматом виробничих приміщень розуміють клімат внутрішнього середовища виробничого приміщання, який визначається поєднаними діями на організм людини, температури, вологості, швидкості руху повітря та теплових випромінювань [23].

Отже, основними параметрами мікроклімату є:

- температура повітря;
- відносна вологість повітря;
- швидкість руху повітря;
- інтенсивність теплового випромінювання;
- температура поверхні.

Параметри мікроклімату можуть змінюватись у широких межах і істотно впливати на самопочуття та здоров'я працівника, продуктивність та якість його праці.

Для робочої зони виробничих приміщень встановлюються оптимальні та допустимі мікрокліматичні умови з урахуванням важкості виконуваної роботи та періоду року згідно з ДСН 3.3.6.042-99 «Санітарні норми мікроклімату виробничих приміщень».

Оскільки розраховано, що на працю в офісі витрачається 87-100 ккал/год, то робимо висновок що вона відноситься до категорії 1а.

Таблиця 4.1 – Допустимі умови мікроклімату для категорії роботи «Легка 1а»

Період року	Температура повітря	Відносна вологість	Швидкість руху
Холодний період року	22 - 24	60 - 40	0,1
Теплий період року	23 - 25	60 - 40	0,1

Перегрівання та переохолодження організму людини веде до порушення життєвих функцій.

Засобами нормалізації параметрів мікроклімату є опалення, вентиляція, кондиціювання повітря, засоби автоматичного контролю і сигналізації тощо. Отож, якщо причиною поганого самопочуття на робочому місці, і як наслідок, низької працездатності є порушення норм мікроклімату в приміщенні, варто звернутися до роботодавця з проханням виправити ситуацію.

Грамотно спроектоване освітлення в офісі - потужний інструмент, що дозволяє в буквальному значенні управляти успішністю співробітників. У будь-якому середньостатистичному офісі працюють співробітники з індивідуальними потребами в освітленості, особливостями зору і персональними умовами праці

Якщо освітлення в офісі організовано на недостатньому рівні, це легко помітити: співробітники стають млявими, дратівливими, вони швидко втомлюються і з небажанням приймаються за роботу [24].

Щоб офісне освітлення працювало на користь треба враховувати наступні параметри:

- колірна температура – для робочих приміщень коливається між нейтральним білим і теплим білим кольором світіння. Ті зони, де важлива гранична концентрація уваги, фахівці рекомендують висвітлювати комбінацією холодних і нейтральних відтінків. Теплий білий колір зробить сприятливою і затишною зону відпочинку;
- світловий потік – залежить від обраного джерела світла і виробника продукції;
- освітлювальна площа – чим вона більша, тим більше уваги слід приділити індивідуальному освітленню робочих місць. Комбінуйте загальне освітлення для офісу з індивідуальними настільними лампами для приміщень на 30 осіб і більше, або використовуйте тільки стельові світильники, якщо в офісі одночасно працюють до 10 осіб;

Освітлення має бути достатнім, щоб можливо було бачити текст і екран, але не настільки яскравим, щоб викликати відблиски або дискомфорт. Для оптимального комфорту та продуктивності слід враховувати наступні фактори:

- кількість світла;
- контраст світла з навколишнім середовищем.

Потенційні небезпеки від кількості світла:

- яскраве світло на екрані дисплея «змиває» зображення, що ускладнює операторам чітке бачення роботи.

Можливі рішення:

- розташуйте ряди вогнів паралельно лінії зору оператора;
- використовуйте регульоване освітлення робочого/настільного столу;

- якщо розсіювачі або альтернативні світильники недоступні, видалення середніх ламп люмінесцентних світильників з 4 лампочками також може зменшити яскравість світла.

Потенційні небезпеки від контраста світла з навколишнім середовищем:

- високий контраст між світлими та темними ділянками екрана комп’ютера, горизонтальною робочою поверхнею та навколишніми ділянками.

Можливе рішення:

- використовуйте світлі, матові кольори та обробку на стінах і стелях, щоб краще відображати непряме освітлення та зменшити темні тіні та контраст.

Шум комп’ютера також має вплив на людину. Оскільки комп’ютер також являється постійним джерелом шуму, то він має негативний вплив на працівника.

Все це знижує працездатність людини і її продуктивність, якість і безпеку праці.Тривала дія інтенсивного шуму (вище 80 дБ) на слух людини приводить до його часткової або повної втрати.

Рівень шуму на робочому місці програмістів не повинен перевищувати 50 дБА, а в залах обробки інформації на обчислювальних машинах – 55 дБА і більше. Робоче місце на підприємстві повино відповідати нормам, так як рівень шуму приблизно 35 – 50 дБА, тому що встановлені комп’ютери з сучасними вентиляторами з мінімальним робочим шумом.

Для зниження рівня шуму використовують добре збалансовані вентилятори, а також проводять технічне обслуговування. Стіни і стеля приміщень, де встановлені комп'ютери, можуть бути облицьовані звукопоглинальними матеріалами.

Електробезпека — система організаційних та технічних заходів і засобів, що забезпечують захист людей від шкідливого та небезпечного впливу електричного струму, електричної дуги, електромагнітного поля і статичної електрики [25].

Великий обсяг електрообладнання в типовому офісі може наражати працівників на серйозну небезпеку електричного струму, включаючи удари, опіки та пожежу.

Електричні аварії, які трапляються в офісному середовищі, зазвичай є результатом несправного або несправного обладнання, небезпечного встановлення або неправильного використання обладнання, зокрема, подовжувачів, роз'ємів живлення та захисних пристроїв від перенапруги.

Для захисту від електричних інцидентів рекомендується:

- навіть при використанні пристрою захисту від перенапруг, переконайтеся, що електричне навантаження не є занадто великим для ланцюга;
- уникайте перевантаження розеток занадто великою кількістю приладів. Ніколи не підключайте одночасно більше одного приладу високої потужності;
- вимикайте прилади з розетки, коли вони не використовуються, щоб заощадити енергію та мінімізувати ризик удару та пожежі;
- перевіряйте електричні шнури один раз на місяць, щоб переконатися, що вони не зношені, тріщини чи інші пошкодження;
- не протягуйте електричні шнури через місця з інтенсивним рухом, під килими або через дверні прорізи;
- подумайте про те, щоб ліцензований електрик встановив додаткові розетки там, де це необхідно, замість того, щоб покладатися на подовжувачі та смужки;
- переконайтеся, що все електричне обладнання сертифіковане національно визнаною лабораторією, і уважно прочитайте всі інструкції виробника.

Випромінювання, що надходить від комп'ютера, відоме як електромагнітне випромінювання надзвичайно низької частоти (ELF). Цей тип випромінювання також випромінюється від ліній електропередач, електричних підстанцій і телебачення.

Випромінювання ELF може викликати або сприяти виникненню різних проблем зі здоров'ям, починаючи від порушення сну та алергічних реакцій до серцевих захворювань, раку та хвороби Альцгеймера. Тривале використання ЕПТ-

екранів вагітними жінками пов’язане з більш високим рівнем викиднів і, можливо, вродженими вадами.

На щастя, існують прості заходи для зменшення радіації комп’ютера:

- випромінювання ELF, природно, спадає дуже швидко зі збільшенням відстані, оскільки випромінювання надходить від джерела низької потужності. Тож просте рішення — розташувати комп’ютер якомога далі за допомогою кабелів (не менше 60 см). Тримайте центральний процесор, принтер та інші пристрої подалі, бажано на підлозі. Це потрібно для того, щоб мінімізувати випромінювання голови та тулуба;
- випромінювання монітора комп’ютера можна уникнути, замінивши ЕПТ-екран на РК та світлодіодні екрани. РК та світлодіодні екрани випромінюють 0,3 мілігауса на відстані 30 см;
- зберігайте принтери на безпечній відстані більше 60 см.

Деякі дротові настільні пристрої особливо нешкідливі, зокрема клавіатури, миші, невеликі колонки, модеми та стаціонарні телефонні лінії. Його можна тримати поблизу людини.

Відповідно до ДСТУ Б В.1.1-36:2016 визначається ступінь вогнестійкості будинків та їх призначення. В комп’ютерному офісі найбільш вірогідні пожежі класів В, горіння твердих речовин, що супроводжується тлінням або самозайманням електроустановок. Для співробітників дуже важливо виконання елементарних правил пожежної безпеки під час перебування на робочому місці. Оскільки безвідповідальне ставлення може спричинити пожежу.

Електричні прилади та машини навколо робочого місця можуть легко стати небезпечними для пожежі. Необхідно перевірте свій офіс на наявність поширених пожежних пасток, та дотримуватись порад щодо запобігання пожежі на робочому місці.

Поради:

- оновлювати комп’ютери, монітори, копіювальні апарати, факсимільні апарати, проектори та подрібнювачі паперу та негайно замінюйте стару або пошкоджену проводку;
- бути обережними у використанні перехідників і не перевантажувати їх;
- якщо у офісі використовуються обігрівачі або інші енергоємні прилади, ніколи не підключanb їх до роз’ємів. Їх слід підключати лише до заземлених розеток, в ідеалі – тих, які не використовуються спільно з іншими штекерами.

Сигналізатори диму – це перша лінія захисту в надзвичайній ситуації. Без належно функціонуючої сигналізації до того часу, коли працівники дізнаються про полум’я, воно вже могло поширитися та стати небезпечним.

Важливе значення має профілактичне обслуговування шляхом оглядів, тестування та технічного обслуговування. На додаток до димових сигналізаторів потрібно створити програму профілактичного технічного обслуговування, котра охоплює всі аспекти системи пожежогасіння та містить належну документацію відповідно до вимог безпеки галузі.

Поради:

- на робочому місці має бути достатня кількість димових сигналізаторів, також треба переконатися, що вони мають джерело живлення та резервну батарею;
- потрібно перевіряти димові сигналізатори щомісяця;
- також регулярно перевіряти всі спринклери та інші системи придушення;
- поєднати свою програму перевірки, тестування та технічного обслуговування з належною документацією, щоб завжди знати, що було перевірено, а що все ще потребує обслуговування.

План евакуації може допомогти спрямувати співробітників, куди звертатися в надзвичайній ситуації. Замість того щоб дозволити страху та хаосу взяти верх, складіть чіткий план зараз і переконайтеся, що ваші співробітники навчені тому, як

його реалізувати. Найкраще запобігання пожежі на робочому місці — це планування.

Потрібно створити план евакуації з кількома маршрутами. Це забезпечить співробітникам кілька можливих виходів на випадок, якщо частина будівлі буде заблокована або недоступна.

Поради:

- обрати місце на безпечній відстані від будівлі, де можуть збиратися евакуйовані працівники;
- призначити одного співробітника на керівну роль, щоб направити людей до місця евакуації і зробити точний підрахунок, коли всі будуть зібрані;
- створити план, котрий підходить для всіх співробітників, незалежно від віку, статі, фізичних чи розумових здібностей.
- зберігати карту маршруту евакуації з усіма виходами, розміщеними на видному місці на робочому місці. Переконайтеся, що він чіткий і легко читається.
- тримати всі номери екстренних служб під рукою. Знадобляться співробітники, щоб мати швидкий доступ до них у надзвичайній ситуації. Обов’язково перевіряйте їх регулярно, щоб переконатися, що інформація все ще точна.

Організація повинна мати в приміщеннях спеціальні засоби:

- кнопки для сповідування о пожежі;
- автоматичні розприскувачі які реагують на дим;
- пожежні крани;
- вогнегасники типу ОУ – 2 чи ОУ – 3.

Усі працівники повинні вміти користуватись вогнегасниками, іншими первинними засобами пожежогасіння, знати місце їх знаходження. Відстань від найбільш віддаленого місця приміщення до місця розташування вогнегасника не повинна перевищувати 20 м. У випадку коли пожежу не вдалося погасити своїми

силами, слід викликати службу пожежної безпеки подзвонивши по номеру «101» [26].

Для безпечної евакуації персоналу поруч з дверними отворами, вимикачами, рубильниками слід розміщувати фотолюмінісцентні евакуаційні знаки.

У разі короткого замикання в одному з блоків ПК або при падінні на підлогу системного блоку, принтера, необхідно відключити ПК від електромережі шляхом від'єднання штекер кабелю електроживлення блоку від розетки, а також доповісти керівнику про те, що трапилося, та викликати фахівця по ремонту комп'ютерів.

Усі фактори були ретельно розглянуті, було приведені рекомендації, щодо зменшення впливу негативних факторів за роботою персонального комп'ютера на підприємстві, основним робочим місцем якого є офіс.

ВИСНОВКИ

В роботі проведено огляд сучасного стану проблеми систем виявлення та захисту веб-додатків від XSS атак за допомогою автоматичного моніторингу веб-логів.

В ході дослідження було обґрунтовано і запропоновано модифікований вид згорткової нейронної мережі (convolutional neural network) для класифікації рукописних цифр MNIST. В якості функції векторизації було запропоновано використовувати шифрування символів в UNICODE формат, замість використання звчиайних для цього функцій. Дана мережа використовується в створеній системі виявлення та протидії XSS атак, де вона класифікує запити в веб-логах сервера.

Використання мережі глибокого навчання полягає в підвищенні ефективності цих самих атак, оскільки минула робота на цю тему використовувала мережу традиційного методу навчання, де найкращим результатом для XSS атаки було лише 91.42 відсотків [14].

Реалізовано власну систему захисту, яка включає такі компоненти:

- класифікатор XSS атак;
- метод захисту від повторної атаки хакерів;
- система виявлення та захисту веб-додатків від XSS.

Створен власний веб-додат для проведеня власних XSS атак та перевірки ефективності створеної системи та її компонентів.

ПЕРЕЛІК ПОСИЛАНЬ

1. What is Cross Site Scripting (XSS). URL: https://www.geeksforgeeks.org/what-is-cross-site-scripting-xss/

2. OWASP Application Security Verification Standard. URL: https://www.owasp.org/index.php/ASVS

3. OWASP Software Assurance Maturity Model. URL: https://www.owasp.org/index.php/SAMM

4. Jovanovic N., Kruegel, C.; Kirda, E. Pixy: A static analysis tool for detecting web application vulnerabilities. *Proceedings of the 2006 IEEE Symposium on Security and Privacy, Berkeley/Oakland, CA, USA*, 2006.

5. Medeiros, I.; Neves, N.; Correia, M. Detecting and removing web application vulnerabilities with static analysis and data mining. *IEEE Trans. Reliab*. 2015, P. 54–69.

6. Sun, F.; Xu, L.; Su, Z. Static Detection of Access Control Vulnerabilities in Web URL: https://www.usenix.org/event/sec11/tech/full_papers/Sun.pdf

7. Medeiros I., Neves N., Correia M. DEKANT: A static analysis tool that learns to detect web application vulnerabilities. *Proceedings of the 25th International Symposium on Software Testing and Analysis, Saarbrücken, Germany, 18–22 July 2016*; P. 1–11.

8. Falana, O.J, Ebo I.O, Tinubu C.O., Adejimi O.A., Ntuk A. Detection of Cross-Site Scripting Attacks using Dynamic Analysis and Fuzzy Inference System. *Proceedings of the 2020 International Conference in Mathematics, Computer Engineering and Computer Science (ICMCECS), Ayobo, Nigeria*, 2020; P. 1–6.

9. Wang R., Xu G., Zeng X., Li X., Feng Z. TT-XSS: A novel taint tracking based dynamic detection framework for DOM Cross-Site Scripting. *J. Parallel Distrib. Comput*. 2018. P. 100–106.

10. Weissbacher M., Robertson W., Kirda E., Kruegel C., Vigna G. Zigzag: Automatically Hardening Web Applications against Client-Side Validation Vulnerabilities.

URL: https://www.usenix.org/system/files/conference/usenixsecurity15/sec15-paper-weissbacher.pdf

11. Duchene F., Rawat S., Richier J.L., Groz R. KameleonFuzz: Evolutionary fuzzing for black-box XSS detection. *Proceedings of the 4th ACM Conference on Data and Application Security and Privacy, San Antonio, TX, USA*. 2014. P. 37–48.

12. Duchene F., Groz R., Rawat S., Richier J.L. XSS vulnerability detection using model inference assisted evolutionary fuzzing. *Proceedings of the 2012 IEEE Fifth International Conference on Software Testing, Verification and Validation, Montreal, QC, Canada*, 2012. P. 815–817.

13. Yirui Wu; Dabao Wei; Jun Feng: Network Attacks Detection Methods Based on Deep Learning Techniques: A Survey URL: https://www.hindawi.com/journals/scn/2020/8872923/

14. Dau Hoang: Detecting Common Web Attacks Based on Machine Learning Using Web Log. *Advances in Engineering Research and Application. ICERA* 2020. Vietnam. 2020.

15. Convolutional Neural Networks, Explained URL: https://towardsdatascience.com/convolutional-neural-networks-explained-9cc5188c4939

16. Fully Connected Layers in Convolutional Neural Networks URL: https://indiantechwarrior.com fully-connected-layers-in-convolutional-neural-networks/

17. How to Develop a CNN for MNIST Handwritten Digit Classification URL: https://machinelearningmastery.com/how-to-develop-a-convolutional-neural-network-from-scratch-for-mnist-handwritten-digit-classification/

18. Get started URL: https://www.jetbrains.com/help/pycharm/quick-start-guide.html#code-assistance

19. Закон України про внесення змін до Кримінального кодексу України щодо підвищення ефективності боротьби з кіберзлочинністю в умовах дії воєнного стану URL: https://zakon.rada.gov.ua/laws/show/2149-20/#Text

20. Закон України про охорону праці URL: https://zakon.rada.gov.ua/laws/show/2694-12#top

21. Безпечні умови URL: http://bo0k.net/index.php?p=achapter&bid=12210 &chapter=1

22. Наказ про затвердження Державних санітарних норм та правил «Гігієнічна класифікація праці за показниками шкідливості та небезпечності факторів виробничого середовища, важкості та напруженості трудового процесу» URL: https://zakon.rada.gov.ua/laws/show/z0472-14#Text

23. Дослідження мікроклімату у виробничих приміщеннях URL: http://opcb.kpi.ua/wp-content/uploads/2014/09/Микроклимат.pdf

24. Освітлення офісу URL: https://www.optimaltd.com.ua/ua/blog/osveshchenie-ofisa/

25. Про заходи з електробезпеки для працівників офісу URL: https://oppb.com.ua/articles/pro-zahody-z-elektrobezpeky-dlya-pracivnykiv-ofisu

26. Про затвердження Правил пожежної безпеки в Україні URL: https://zakon.rada.gov.ua/laws/show/z0252-15#Text

Додаток А. Лістинг програмного продукту

Код програми тренування та тестування

```
import pandas as pd
import numpy as np
from tensorflow import keras
from tkinter import *
import re

def generate_dataset(df):
    tensorlist = []
    labelist = []
    maxabs = 0
    for row in df.values:

        query = row[0]
        label = row[1]

        url_part = [ord(x) if (ord(x) < 129) else 0 for x in
query[:784]]
        url_part += [0] * (784 - len(url_part))

        maxim = max(url_part)
        if maxim > maxabs:
            maxabs = maxim
        x = np.array(url_part).reshape(28, 28)

        if label == 1:
            y = np.array([0, 1], dtype=np.int8)
        else:
            y = np.array([1, 0], dtype=np.int8)
        tensorlist.append(x)
        labelist.append(y)

    return tensorlist, labelist
def collect():
    textfile = open('C:\\Apache24\\logs\\error.log', 'r')
    filetext = textfile.read()
    textfile.close()

    matches_1 = re.findall("\[client (.*):[0-9]*] .* POST", filetext)
    matches_2 = re.findall("csrfmiddlewaretoken=.*&text=(.*)",
filetext)

    print(matches_1)

    for x in range(len(matches_2)):
```

```
        if "+" in matches_2[x]:
            matches_2[x] = matches_2[x].replace('+', ' ')

    print(matches_2)

    #matches = {matches_2[i]: matches_1[i] for i in
range(len(matches_2))}
    #print(matches)

    with open('sample.txt', 'w') as the_file:
        for x in matches_2:
            the_file.write(f'{x}\n')

    with open('sample_id.txt', 'w') as the_file:
        for x in matches_1:
            the_file.write(f'{x}\n')
def work():

    post = ''

    textfile = open('C:\\Apache24\\logs\\error.log', 'r')
    filetext = textfile.read()
    textfile.close()

    matches_1 = re.findall("\[client (.*):[0-9]*] .* POST", filetext)
    matches_2 = re.findall("csrfmiddlewaretoken=.*&text=(.*)",
filetext)

    print(matches_1)

    for x in range(len(matches_2)):
        if "+" in matches_2[x]:
            matches_2[x] = matches_2[x].replace('+', ' ')
        post += f'{matches_2[x]}, {matches_1[x]}\n'

    print(matches_2)

    checker,blocklist = [],[]
    model = keras.models.load_model('model.h5')

    f =
open('C:\\Users\\alexp\\PycharmProjects\\Diplom\\Vul\\sample.txt',
encoding='utf-8', mode='r')
    #f =
open('C:\\Users\\alexp\\PycharmProjects\\Diplom\\data\\goodqueries12000
00.txt', encoding='utf-8', mode='r')
    allqueries = f.readlines()
    f.close()
```

```
    allqueriespd = pd.DataFrame({'query': allqueries, 'label': [2 for x
in allqueries]})
    print(allqueriespd)

    X, y = generate_dataset(allqueriespd)

    prediction = model.predict(np.array(X)/128.0)

    pred = prediction.tolist()

    print(pred)

    amount = 0

    for x in range(len(pred)):
        if pred[x][0] > 0.5:
            amount += 1
            checker.append(0)
        else:
            checker.append(1)

    print(checker)

    with
open('C:\\Users\\alexp\\PycharmProjects\\Diplom\\Vul\\sample_id.txt')
as f:
        lines = f.read().splitlines()

    print(lines)

    for x in range(len(lines)):
        if checker[x] == 1:
            blocklist.append(lines[x])

    blocklist = list(dict.fromkeys(blocklist))

    print(blocklist)

    with open('C:\\Apache24\\conf\\IPList.conf', 'w') as the_file:
        for x in blocklist:
            the_file.write(f'Require not ip {x}\n')

    for x in pred:
        post += f"{str(x)}\n"

    post += f'Amount of normal requests {amount}\n'
    post += f"Amount of abnormal requests {len(pred)- amount}\n"

    post += f"Blacklist: {str(blocklist)}"
```

```
    T.insert(END, post)

root = Tk()
root.title("Diploma")
root.geometry("625x325")

Activator = Button(text="Check", font="Arial 16 bold", command=work)
Activator.place(x=280, y=20)

T = Text(root, height=9, width=45, font="Arial 16 bold")
T.place(x=30, y=80)

root.mainloop()
```

Код програми виявлення та протидії XSS атаки

```
import pandas as pd
import numpy as np
from keras.models import Sequential,Model
from keras.layers import Dense, Conv2D, Flatten, MaxPool2D
from mlxtend.plotting import plot_confusion_matrix
from keras import layers
from tensorflow import keras
from keras import metrics
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix
import matplotlib.pyplot as plt
from tkinter import *
from tkinter import filedialog as fd

def generate_dataset(df):
    tensorlist = []
    labelist = []
    maxabs = 0
    for row in df.values:

        query = row[0]
        label = row[1]

        url_part = [ord(x) if (ord(x) < 129) else 0 for x in
query[:784]]
        url_part += [0] * (784 - len(url_part))

        maxim = max(url_part)
        if maxim > maxabs:
            maxabs = maxim
```

```
        x = np.array(url_part).reshape(28, 28)

        if label == 1:
            y = np.array([0, 1], dtype=np.int8)
        else:
            y = np.array([1, 0], dtype=np.int8)
        tensorlist.append(x)
        labelist.append(y)

    return tensorlist, labelist
def train(event):
    p_1,p_2 = path_1['text'], path_2['text']

    f = open(p_1, encoding='utf-8', mode='r')
    badqueries = f.readlines()
    f = open(p_2,encoding='utf-8',mode='r')
    goodqueries = f.readlines()
    f.close()

    badqueriespd = pd.DataFrame({'query': badqueries,'label': [1 for x
in badqueries]})
    goodqueries = pd.DataFrame({'query': goodqueries,'label': [0 for x
in goodqueries]})
    allqueries = pd.concat([badqueriespd,goodqueries])

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

    model = Sequential()
    model.add(layers.Input(shape=(28,28,1)))
    model.add(Conv2D(64,       (5,       5),       activation='relu',
kernel_initializer='he_uniform'))
    model.add(layers.BatchNormalization())
    model.add(Conv2D(32, (5, 5), activation='relu'))
    model.add(layers.Dropout(0.2))
    model.add(MaxPool2D((3,3)))
    model.add(Flatten())
    model.add(Dense(100,                                activation='relu',
kernel_initializer='he_uniform'))
    model.add(Dense(2, activation='softmax'))
    model.compile(optimizer='adam',    loss='categorical_crossentropy',
metrics=['accuracy',metrics.Recall(),metrics.Precision()])

    print(len(X_train))
    print(len(y_train))
    print(model.summary)
```

```
    history                                                                 =
model.fit(np.array(X_train)/128.0,np.array(y_train),epochs=3,batch_size
=100)
    #model.evaluate(np.array(X_test)/128.0,np.array(y_test))
    print('aee')

    #model.save('model4.h5')
    a.delete('1.0', END)
    a.insert(END, 'List of accuracy per epoch:\n')
    a.insert(END, history.history["accuracy"])


    y_pred = model.predict(np.array(X_test)/128.0)
    classes_y = np.argmax(y_pred, axis=1)



    new_fuck = []
    fuck = np.array(y_test).tolist()
    for x in fuck:
        if x[0] == 1:
            new_fuck.append(0)
        else:
            new_fuck.append(1)

    #print(np.array(new_fuck))

    cm = confusion_matrix(np.array(new_fuck), classes_y)
    print(cm)
    plot_confusion_matrix(conf_mat=cm)
    plt.show()
def plot(history):

    plt.plot(history.history["accuracy"])
    plt.title('Model accuracy')
    plt.ylabel('Accuracy')
    plt.xlabel('Epoch')
    plt.legend(['Train', 'Test'], loc="upper left")
    plt.show()

    plt.plot(history.history['loss'])
    plt.title("Model loss")
    plt.ylabel("Loss")
    plt.xlabel("Epoch")
    plt.legend(["Train", "Test"], loc="upper left")
    plt.show()
def Destiny_1(event):
    path1 = fd.askopenfilename()
    path_1['text']=path1
```

```
def Destiny_2(event):
    path2 = fd.askopenfilename()
    path_2['text']=path2
def Destiny_3(event):
    path3 = fd.askopenfilename()
    path_3['text']=path3
def test(event):

    ex = path_3['text']
    p_1 = path_1['text']

    model = keras.models.load_model(ex)

    f = open(p_1, encoding='utf-8', mode='r')
    allqueries = f.readlines()
    f.close()

    allqueriespd = pd.DataFrame({'query': allqueries, 'label': [2 for x
in allqueries]})
    print(allqueriespd)

    X, y = generate_dataset(allqueriespd)

    prediction = model.predict(np.array(X) / 128.0)

    pred = prediction.tolist()

    print(pred)

    model.summary()

    amount = 0

    for x in range(len(pred)):
        if pred[x][0] > 0.5:
            amount += 1

    print(f'Amount of normal requests {amount}\n')
    print(f"Amount of abnormal requests {len(pred) - amount}\n")

    a.delete('1.0', END)
    a.insert(END, f'Amount of normal requests {amount}\n')
    a.insert(END, f'Amount of normal requests {len(pred) - amount}\n')

root = Tk()
root.title("Diploma")
root.geometry("625x325")

a = Text(root, bg='black', fg='white')
```

```
mainmenu=Menu(root)

choice_1=Button(text='Choose malicious samples')
choice_1.bind('<Button-1>',Destiny_1)
choice_1.pack(fill = X)

path_1 = Label(root)
path_1.pack(fill = X)

choice_2=Button(text='Choose benign samples')
choice_2.bind('<Button-1>',Destiny_2)
choice_2.pack(fill = X)

path_2 = Label(root)
path_2.pack(fill = X)

bruh_1=Button(text='Train')
bruh_1.bind('<Button-1>',train)
bruh_1.pack(fill = X)

choice_3=Button(text='Choose desire model')
choice_3.bind('<Button-1>',Destiny_3)
choice_3.pack(fill = X)

path_3 = Label(root)
path_3.pack(fill = X)

bruh_2=Button(text='Test')
bruh_2.bind('<Button-1>',test)
bruh_2.pack(fill = X)

a.pack(fill=X)

root.mainloop()
```