Міністерство освіти і науки України
Київський національний університет імені Тараса Шевченка

Факультет інформаційних технологій
Кафедра кібербезпеки та захисту інформації

ПОЯСНЮВАЛЬНА ЗАПИСКА

кваліфікаційної роботи магістра

галузь знань *12 Інформаційні технології*
(шифр і назва галузі знань)

спеціальність *125 Кібербезпека*
(код і назва спеціальності)

освітній ступень *магістр*

освітньо-наукова програма *Кібербезпека*
(назва освітньої програми)

на тему: «Моделі ідентифікації вразливостей у веб застосунках»

Виконавець: студент ІІ курсу, групи КБм-21

(підпис) Шайна Олексій Максимович
(Ім'я, ПРІЗВИЩЕ)

	Ім'я, ПРІЗВИЩЕ	**Підпис**
Науковий керівник	Тетяна БАБЕНКО	
Нормоконтроль	Сергій ДАКОВ	

Київ 2023

Міністерство освіти і науки України
Київський національний університет імені Тараса Шевченка

Факультет інформаційних технологій
Кафедра кібербезпеки та захисту інформації

ЗАТВЕРДЖЕНО:
В.о. завідувача кафедри
кібербезпеки
та захисту інформації
_____ Сергій ТОЛЮПА
«24» жовтня 2022 р.

ЗАВДАННЯ
на виконання кваліфікаційної роботи

спеціальності *125 Кібербезпека*
(код і назва спеціальності)

освітній ступень *магістр*

Здобувача(ки) КБм-21 (група) Шайни Олексія Максимовича (прізвище ім'я по-батькові)

Тема дипломної роботи ***Моделі ідентифікації вразливостей у веб застосунках***

1. ПІДСТАВИ ДЛЯ ПРОВЕДЕННЯ РОБОТИ

Рішення засідання кафедри кібербезпеки та захисту інформації факультету інформаційних технологій протокол № 3 від 20.10.2022

2. МЕТА ТА ВИХІДНІ ДАНІ ДЛЯ ПРОВЕДЕННЯ РОБІТ

Об'єкт досліджень	Процес забезпечення кібербезпеки веб застосунків
Предмет досліджень	Послуга безпеки у веб застосунках
Мета	Розробка моделі ідентифікації вразливостей у веб застосунках
Вихідні дані для проведення роботи	Методи ідентифікації вразливостей у веб застосунках

3. ОЧІКУВАНІ НАУКОВІ РЕЗУЛЬТАТИ

Наукова новизна удосконалення процесу ідентифікації вразливостей у веб застосунках

Практична цінність покращення моделі ідентифікації вразливостей у веб застосунках.

4. ВИМОГИ ДО РЕЗУЛЬТАТІВ ВИКОНАННЯ РОБОТИ

Робота виконана у повному обсязі відповідно до теми.

5. ЕТАПИ ВИКОНАННЯ РОБОТИ

Найменування етапів робіт	Строки виконання робіт (початок-кінець)
Розробка плану для досягнення мети роботи	24.10.2022 – 23.01.2022
Аналіз літературних джерел	24.01.2022 – 14.02.2022
Розробка моделі ідентифікації вразливостей у веб застосунках	15.02.2022 – 24.04.2022
Оформлення і друк пояснювальної записки	25.04.2022 – 19.05.2023

6. РЕАЛІЗАЦІЯ РЕЗУЛЬТАТІВ ТА ЕФЕКТИВНІСТЬ

Економічний ефект Зниження збитків через інциденти кібербезпеки

Соціальний ефект Покращення технологій реагування на інциденти кібербезпеки

7. ДОДАТКОВІ ВИМОГИ

Завдання видав _____ Тетяна БАБЕНКО
(підпис) (Ім'я, ПРІЗВИЩЕ)

Завдання прийняв до виконання _____ Олексій ШАЙНА
(підпис) (Ім'я, ПРІЗВИЩЕ)

Дата видачі завдання: 24.10.2022 р.
Термін подання дипломної роботи до ЕК 19.05.2023 р.

РЕФЕРАТ

Дипломна робота складається зі вступу, трьох розділів, загальних висновків, списку використаних джерел, додатків, має 70 сторінки основного тексту. Список використаних джерел містить 107 найменування і займає 8 сторінок.

Об’єктом дослідження в данній роботі є процес забезпечення кібербезпеки web-застосунків.

Метою данної роботи є розробка моделі ідентифікації вразливостей у веб застосунках.

Предметом дослідження в данній роботі - це послуга безпеки у веб застосунках.

Методи дослідження дипломної роботи:

- проектування нейронних мереж;
- математичної статистики
- побудови дерев рішень
- прийняття рішень
- компютерне моделювання

У роботі проведено дослідження найрозповсюдженіших вразливостей у веб застосунках. Проведено аналіз подій, що відбуваються у веб застосунках. Реалізовано модель ідентифікації вразливостей у веб застосунках, з використанням нейронної мережі.

Практична цінність полягає у тому, що реалізвоану модель можна використовувати як послугу безпеки для веб застосунків.

Зпроектована модель може використовуватися фахівцями з кібербезпеки, які виконують аналіз інцидентів.

Наукова новизна: вперше запропонована послуга безпеки, що дозволяє виявляти та ідентифік

Актуальність теми дипломної роботи визначається тим, що веб застосунки є основною складовою використання сервісу інтернет. Загрози що трапляються в додатках не виняткові, і задля мінімазії шкоди при атаках, використання моделі ідентифікації вразливостей сприяє більш швидкому реагуванню на інциденти, що знижує принципову шкоду при кібер атаках.

Ключові слова: веб застосунок, кібербезпека, broken-access-control, injection, server-side request foriegery

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ ТА СКОРОЧЕНЬ

ПЗ	–	Програмне Забезпечення
SSRF	–	Server-Side Request Forgery
CMD	–	Command Line
XSS	–	Cross-Site Scripting
CLF	–	Common Log Format
CLF	–	Combined Log Format
SSTI	–	Server-Side Template Injection

ЗМІСТ

ВСТУП

Актуальність данної роботи визначається тим, що користувачі використовують веб застосунки кожен день, діляться персональною інформацію у мережі за допомогою веб-сервісів. Тому захист від розкриття конфіденційних даних під час використання веб ресурсів є важливим.

Інформаційні технології стали невід'ємною частиною життя людини. Кожен день користувачі всесвітньої павутини обмінюються мільйонами мегабайтів інформації через мережу. Ми використовуємо мобільні додатки для спілкування одне з одним, створюємо особисті кабінети на різних веб-порталах, довіряємо свої персональні данні веб застосункам: номера мобільних телефонів та кредитних карток, паспортні данні. Використовуючи мережу кожного дня, з'являється питання, чи можна довіряти конфіденційні данні, передаючи їх веб-додаткам та іншим користувачам всесвітньої павутини. З різким зростанням інформаційного простору, зросла й кількість різноманітних видів атак на мережу та веб-ресурси. Компанія OWASP постійно оновлює рейтинг потенційних та найпопулярніших атак у веб застосунках.

Держави з всього світу сформували певні вимоги та стандарти, які затверджують системи захисту та найкращі практики з безпеки веб-додатків. Однією з вимог є реагування на інциденти у веб застосунках

Існує не так багато моделей для ідентифікації вразливостей веб-додатків на момент виявлення вразливостей. Найпопулярнішою є Threat Modeling. Це модель тестування, яку можна використовувати, щоб допомогти вам визначити загрози, атаки, уразливості та контрзаходи, які можуть вплинути на ваш веб застосунок. Ви можете використовувати моделювання загроз, щоб сформувати дизайн програми, досягти цілей безпеки вашої компанії та зменшити ризик.

Тому метою роботи є проектування моделі ідентифікації вразливостей у веб застосунках.

Об’єктом дослідження в данній роботі є процес забезпечення кібербезпеки web-застосунків.

Предметом дослідження в данній роботі - це послуга безпеки у веб застосунках.

Методи дослідження дипломної роботи:

- проектування нейронних мереж;
- математичної статистики
- побудови дерев рішень
- прийняття рішень
- компютерне моделювання

Апробація результатів роботи та публікації за темою кваліфікаційної роботи:

1. Олексій Шайна, Тетяна Бабенко Моделі ідентифікації вразливостей у веб застосунках. Матеріали VI Міжнародній науково-практичній конференції "Проблеми кібербезпеки інформаційно-телекомунікаційних систем (PCSITS)" (Київ, 2023).

.

РОЗДІЛ 1
АНАЛІЗ ІСНУЮЧИХ АРХІТЕКТУР ТА ЗАГРОЗ У ВЕБ ЗАСТОСУНКАХ

1.1 Архітектура веб додатків

Архітектура веб-додатків — це «скелет» або макет, який відображає взаємодію між компонентами додатків, системами проміжного програмного забезпечення, інтерфейсами користувача та базами даних. Така взаємодія дозволяє кільком додаткам працювати разом одночасних [1]. На рисунку 1.1 зображено схему за якою відбуваються комунікації компонентів веб застосунку[2]. Умовно веб архітектуру можна поділити на 2 основні складові – Frontend, Backend [3].

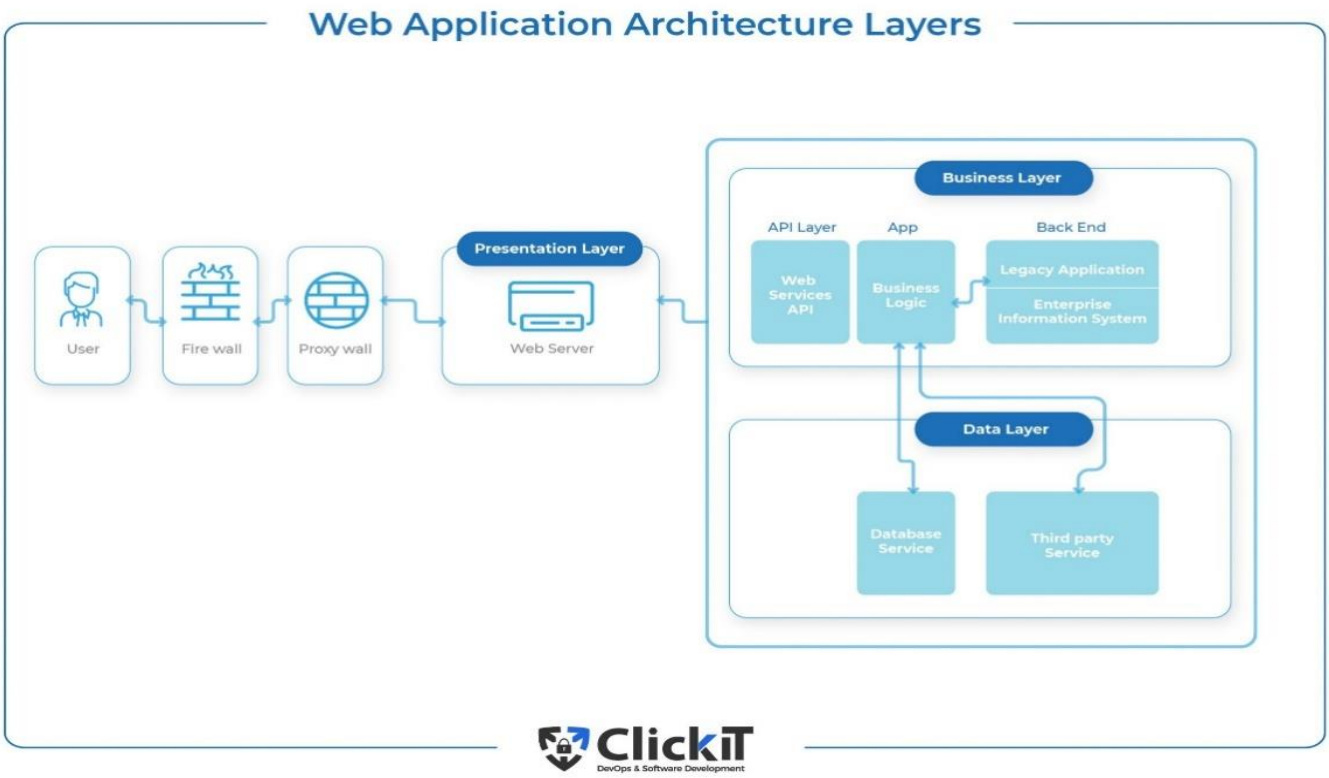

Рисунок 1.1 Архітектура веб застосунків

Frontend – це публічна частина веб додатку, з якою користувач може взаємодіяти і контактувати напряму [10]. Саме тут відбувається відображення

функціональних завдань, призначеного для користувача інтерфейсу, що виконується на стороні клієнта, а також обробка запитів користувачів (Рис 1.2) [4]. Вразливості, які існують в цій частині архітектурі не виняткові та трапляються досить часто [7].

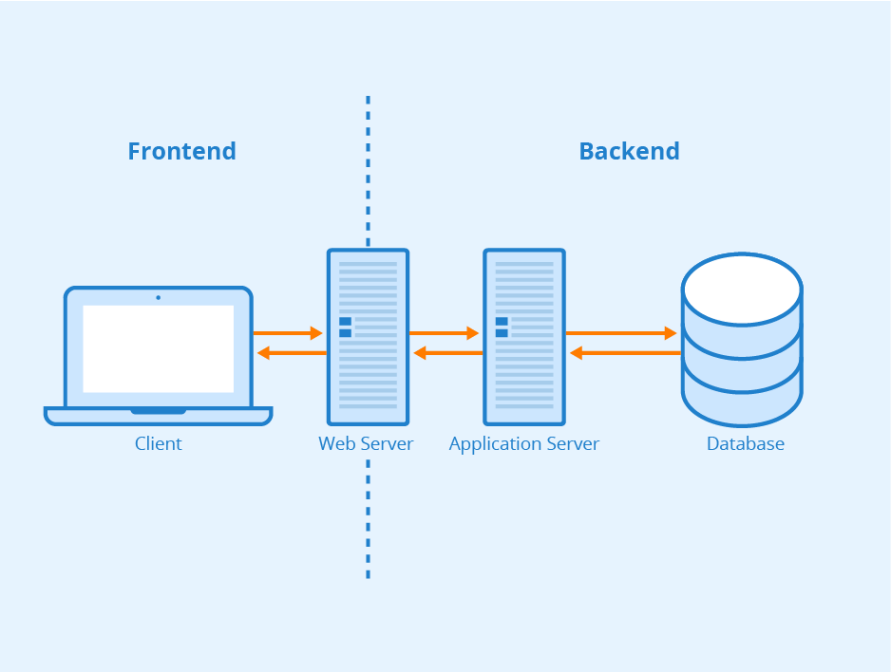

Рисунок 1.2 Схема Frontend та Backend

Backend – це серверний бік будь-якого сайту або програми, яка відповідає за всю, логіку яка відбувається в додатку [11]. Всі ці процеси ми фактично не бачимо, тому й він і називається прихованою стороною продукту (Рис 1.2) [12].

Взаємодія цих обох частин архітектури відбувається по колу: клієнтська частина програми відправляє інформацію на сервер (backend), надалі програма на сервері опрацьовує інформацію та після цього повертає клієнтській стороні зрозумілий для користувача формі [14].

Невід'ємною частиною взаємодії клієнтської та серверної частин є користувач. Саме користувач робить HTTP запити до серверу, через клієнтську частину (браузер) і саме через те, що користувач контролює данні, які відправляються до серверу існує велика кількість різних вразливостей [15].

1.2 Проект з безпеки веб застосунків

Для виявлення вразливостей у веб-додатках фахівець з безпеки повинен розумітися на більш вразливих місцях та одразу сконцентрувати увагу на low hanging fruit (Найбільш потенційні вразливості) [9]. Ці типи вразливостей мають найбільш суттєвий вплив на веб-додатки, наприклад SSTI, SQL або Broken Access Control.

OWASP – відкритий проект з безпеки веб застосунків, постійно оновлює список найбільш розповсюдженіших вразливостей у додатках [22]. Ця компанія не пов’язана з жодним комерційним проектом і являє собою суспільство, яке складається з фахівців в області інформаційної безпеки. Цей проект з певною періодичністю оновлює лист з найбільш потенційних вразливостей, наприклад OWASP TOP 10 опублікував наступні 10 найпоширеніших вразливостей на момент 2017 року (Рис 1.4) [23].

Рисунок 1.3 OWASP TOP 10 2017

У 2017 році як ми бачимо найпоширенішою вразливістю була ін’єкція, такі як SQL, SSTI, CMD [24]. Приблизно раз на 4 роки цей список оновлюється, підводячи підсумки, які саме вразливості зустрічалися найчастіше. У 2021 році проект опублікував оновлений лист low hanging fruits (Рис 1.4) [26].

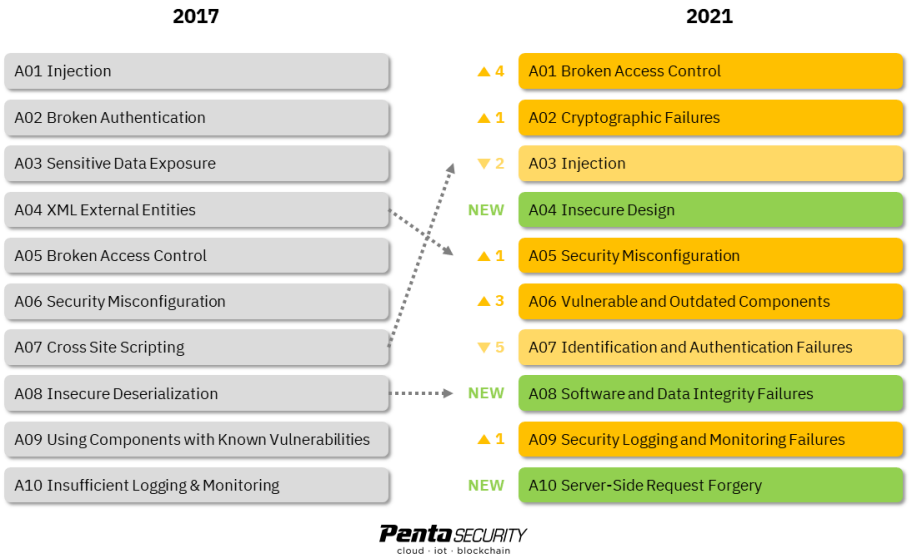

Рисунок 1.4 OWASP TOP 10 2017

Як можна побачити на рисунку 1.4, багато чого змінилося: вразливість Broken Access Control почала займати перше місце серед усіх інших, а ін’єкція почала займати 3 місце. Деякі типи вразливостей було перейменовано, але це не суттєво. Взагалі OWASP TOP 10 – є визнаною світовою методологією оцінки вразливостей у веб застосунках у цілому світу і відображає сучасні тренди безпеки веб-додатків, також є першим кроком організації до створення культури більш безпечного програмного забезпечення [27].

З кожним роком кількість веб-атак пропорційно зростає [28]. З розробкою все більшої кількості програмного забезпечення та легшої організації атак у 2022 році відбулося приблизно наступне:

1.Приблизно 236 мільйонів атак на веб застосунки, лише у першій половині року.

2.Велика Британія посягала перше місце серед країн з найбільшою кількістю жертв від кібератак: на 1 мільйон 4783 у листопаді на 1 мільйон користувачів, що на 40 відсотків більше ніж минулого року.

3.76 відсотків власників веб-додатків у більше ніж 70 країн світу, зазначили, що відбулася хоча б 1 атака на їх веб-сайт, а це на 50 відсотків більше аніж минулого року.

4.70 відсотків власників середнього та малого бізнесів побоюються, що залишаться без грошей, після однієї вдалої атаки.

5.Найпоширенішою атакою з якою зіткалися власники компаній це фішинг, не менш поширеною була атака на веб застосунки, за допомогою виявлення потенційних вразливостей.

6.Приблизна середня вартість витоку персональної інформації про фінансові послуги одна з найбільших серед інших галузей і становить 5.85 мільйонів доларів у середньому.

7.Компаніям, що займаються фінансами потрібно приблизно 233 дні в середньому, щоб локалізувати атаку та мінімізувати ризики з витоку інформації.

Одна з найгучніших атак, яка відбулася наприкінці 2021 року, це була атака log4j. Exploit Log4j почався як одна вразливість, але став низкою проблем, пов’язаних з Log4j та інтерфейсом Java Naming and Directory Interface (JNDI), що є основною причиною exploit. За офіційними публічними даними ця вразливість вплинула на велику кількість різноманітних сервісів та веб застосунків, що використовували цю бібліотеку для логування, такі як: веб-сервер Apache, Apple iCloud Service, VMware products та багато інших. Причиною атак такого типу на веб-додатки, є використання вразливого ПЗ. За шкалою OWASP TOP 10 2021, такий тип атаки посягає 6 місце по розповсюдженості коли застосунок використовує не оновленні компоненти.

1.3 Основні типи тестування захищеності веб додатків

Концепція білої скриньки.

Цей метод використовує тип тестування білої скриньки, коли тестувальник має доступ до програмного забезпечення і може почати аналізувати потенційні вразливості з ПЗ. В теорії даний тип тестування дозволяє виявити якомога більше потенційних вразливостей до релізу. Сучасні веб-фреймворки запобігають більшій частині найбільш розповсюджених вразливостей, проте вони не можуть повністю зробити додаток безпечним від атак. Однією з найкращих практик – це звісно використання внутрішніх інструментів для розробки, представлені самим фреймворком. Так як це програмне забезпечення більш перевірене та має відкритий код це підвищує рівень його стійкості перед типовими атаками зловмисників. В ідеальній концепції білий список повинен створюватися під час розробки застосунку. Реалізація білого списку можлива при створені веб застосунку використовуючи будь-який фреймворк на основі MVC так як одна з особливостей роботи таких фреймворків має у собі механізм який працює за концепцією посередника і відповідає за перехоплення HTTP запиту і видачу дозволеної відповіді, яка задається відповідною моделлю білого списку (Рис 1.5).

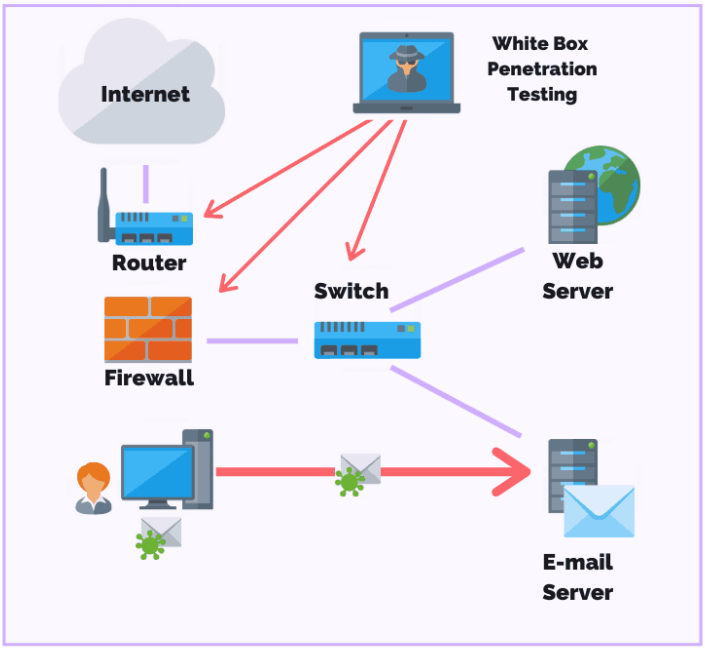

Рисунок 1.5 Тестування білим ящиком

DAST

Цей метод використовує тип тестування чорної скриньки, коли тестувальник має лише доступ до веб-додатку, і починає тестування, не знаючи про інфраструктуру додатку рівно нічого. Даний метод тестування має свій сенс коли хочеться зрозуміти, як виглядає застосунок з точки зору хакеру. Не маючи доступу до коду да знань стосовно архітектури надає атакуючому місце для фантазії. Використовувати даний метод для покриття якомога більшої кількості вразливостей не бажано, так як багато часу потрібно витратити саме на збір інформації про ціль.

Source Code Review

Цей метод логічно можна було б віднести до складової SAST, але це не зовсім так. Річ у тому що сучасні веб застосунки написані з використанням мови JavaScript, яку браузер використовує для відтворення елементів у зрозумілий для користувача формат. Фактично клієнт має доступ до цього коду, як і браузер, тому аналіз цієї

ділянки є також дуже важливим. Можна сказати що завдяки перевірці коду, можливо виявити дуже багато потенційних вразливостей ще на стадії розробки. Тому нерідко цей метод тестування використовується у CI/CD процесі.

1.4 Основні типи загроз у веб застосунках

Вразливості у веб застосунках є найбільшими загрозами, для того щоб розуміти як їх можливо усунути або не допускати зовсім, компанія OWASP, розробила спеціальну класифікацію для всіх типів загроз [26]. Найбільш розповсюдженою класифікацією є рейтинг OWASP TOP 10. Саме цьому рейтингу потрібно приділити найбільшу увагу [27].

1.4.1 Broken Access Control

Дана вразливість виникає коли при запиті якогось об’єкта у веб застосунку, немає необхідної перевірки доступу [29][30]. Наприклад це може бути адмін панель або профіль будь-якого користувача в системі [31]. Застосунок, перш ніж відобразити сторінку клієнту, повинен перевірити доступ клієнта до цього об’єкта. До цієї вразливості відносяться також: IDOR, Directory Traversal [32].

1.4.2 Cryptographic Failures (Sensitive data exposure)

Вразливість цього типу виникає коли при розробці, у коді або елементі сторінки кодується приватна інформація, така як токени доступу до ресурсів, паролі тощо [35]. Увага в цьому зосереджується більше на причині виникання цієї вразливості, перш за все пов’язаної з криптографією або її відсутністю [32][34]. До відомих загальних недоліків входять CWE 259 використання закодованого паролю, CWE 327 зламаний або ризикований криптографічний алгоритм, CWE 331 недостатня ентропія [33][36].

1.4.3 Injections

Цей тип вразливості виникає коли, дані, які передаються клієнтом не перевіряються належним чином, тобто не проходять обробку перш ніж передати корисне навантаження до серверу [38]. Без використання екранування та динамічних запитів або непараметеризованих викликів, або коли данні клієнта використовуються в параметрах пошуку об’єктно реляційного відображення (ORM) для отримання даних після запиту [39]. При цій атаці ворожі дані використовуються безпосередньо або об’єднуються. SQL або команда містить структуру та шкідливі дані в динамічних запитах, командах або збережених процедурах [40][41].

1.4.4 Insecure Design

Незахищений дизайн – дана категорія, відображає різні слабкі сторони, представлені як – відсутній або неефективний дизайн контролю [47]. Дана атака не є джерелом для всіх інших категорій з OWASP TOP 10 [48]. Значна різниця між незахищеним дизайном та незахищеним впровадженням[49]. Розрізняються недоліки дизайну веб застосунку та способи їх впровадження, так як вони мають різні першопричини та способи їх запобігання. Захищений дизайн може мати дефекти реалізації, що приводить до потенційних вразливостей [50]. А ось незахищений дизайн не можна виправити ідеальною реалізацією, оскільки за визначенням засоби контролю безпеки не створювалися для даного типу дизайну. Виникання даної вразливості трапляється через відсутність профілювання бізнес-ризиків програмного забезпечення або системи, а отже при цій умові неможливо впровадити достатній контроль за об’єктами веб застосунку, які є дійсно важливими для додатку [51].

1.4.5 Security Misconfiguration

Дана вразливість виникає коли відсутнє підсилення захисту безпеки в будь-якій частині стеку програми або невірно налаштований доступ до хмарних служб [55]. При використанні або встановленні непотрібних функцій, такі як відкриті порти або інстальоване стороннє програмне забезпечення а також при некоректно виставлених привілеях – можна проексплуатувати сервіс або службу не так як задумано [55]. До цього також відносяться облікові записи за замовчуванням і їх паролі залишаються активними і не змінюються [53]. Увімкнена обробка помилок також дає користувачу дуже багато інформації про систему [52][57].

1.4.6 Vulnerable and Outdated Components

Вразливість використання застарілого програмного забезпечення або якогось з компонентів системи, який потребує оновлення виникає при недостатньому моніторингу вразливостей для пз, яке використовується у веб застосунку [60][61]. Зазвичай найкращою практикою є постійне оновлення до останньої версії бібліотеки або фреймворку, але іноді це не є можливим [62]. В цьому випадку необхідно оцінити ризики використання вразливого пз [63]. Наприклад якщо використовується фреймворк, в якому вразлива панель адміністратора до виконання шкідливого коду на стороні серверу, в цьому випадку потрібно оцінити ризик, при якому атакуючий може підвищити привілеї від звичайного користувача до admin, або взагалі отримати доступ до панелі [65]. Веб застосунки з вразливими бібліотеками або компонентами трапляються у 80 відсотках випадках [66].

1.4.7 Identification and Authentication Failures

Підтвердження особистості користувача є одним з найважливіших компонентів захисту [32]. Нерідко трапляються недоліки автентифікації, якщо веб застосунок дозволяє атаки перебору паролів або інші автоматизовані дії на стороні серверу [68]. До цієї категорії також відносяться: дозвіл використання слабких паролів, які не відповідають політиці стійких паролів, використання неефективних процесів

відновлення паролів, наприклад відповіді на основі знань, які неможливо зробити безпечними [69]. Сюди також можна віднести використання слабких хешів або відсутність багатофакторної автентифікації [35]. Нерідкістю є повторне використання ідентифікаторів сеансу cookie, після успішного входу у веб застосунок [67].

1.4.8 Software and Data Integrity Failures

Порушення цілісності програмного забезпечення та даних стосуються коду та інфраструктури, які не захищають від порушень цілісності [71]. Прикладом цього є ситуація, коли програма використовує плагіни, бібліотеки або модулі з ненадійних джерел, сховищ і мереж доставки вмісту (CDN). Незахищений конвеєр CI/CD може призвести до несанкціонованого доступу, зловмисного коду або компрометації системи [75]. Нарешті, багато програм тепер включають функцію автоматичного оновлення, коли оновлення завантажуються без достатньої перевірки цілісності та застосовуються до попередньо довіреної програми [72]. Зловмисники потенційно можуть завантажити власні оновлення для розповсюдження та запуску на всіх інсталяціях [73]. Іншим прикладом є те, що об’єкти чи дані кодуються або серіалізуються в структуру, яку зловмисник може побачити та змінити, вразливу до незахищеної десеріалізації [74].

1.4.9 Security Logging and Monitoring Failures

Ця категорія має допомогти виявляти, розголошувати та реагувати на активні порушення [76]. Без реєстрації та моніторингу порушення неможливо виявити. Недостатнє ведення журналу, виявлення, моніторингу та активної відповіді відбувається будь-коли: події, які підлягають аудиту, як-от входи, невдалі входи та транзакції великої вартості, не реєструються [77]. Попередження та помилки не створюють жодних, неадекватних або незрозумілих повідомлень журналу. Журнали програм і API не перевіряються на наявність підозрілої активності. Журнали зберігаються лише локально. Відповідні порогові значення сповіщень і процеси

ескалації відповіді відсутні або не діють. Тестування на проникнення та сканування інструментами динамічного тестування безпеки додатків (DAST) (такими як OWASP ZAP) не викликають попереджень [79]. Програма не може виявляти, ескалувати або сповіщати про активні атаки в режимі реального часу або майже в реальному часі [78].

1.4.10 Server-Side Request Forgery

Помилки SSRF виникають щоразу, коли веб-програма отримує віддалений ресурс без перевірки наданої користувачем URL-адреси [80]. Це дозволяє зловмиснику змусити програму надіслати створений запит до неочікуваного адресата, навіть якщо він захищений брандмауером, VPN або іншим типом списку контролю доступу до мережі (ACL) [82]. Оскільки сучасні веб-додатки надають кінцевим користувачам зручні функції, отримання URL-адреси стає звичайним сценарієм [79]. Як наслідок, захворюваність на SSRF зростає [81]. Крім того, серйозність SSRF стає вищою через хмарні сервіси та складність архітектур [83].

1.4.11 XSS

Дана категорія атаки виникає коли на сторінки, згенерована за допомогою серверу, потрапляють або виконуються клієнтські javascript корисні навантаження [85]. Сенс атаки у тому що ця атака безпосередньо концентрована на сторону клієнта, тобто основною ціллю атакуючого є інший клієнт серверу. Гарно згенерований сценарій дозволяє вкрасти сесію користувача, а якщо атака автоматизована то й взагалі у групи користувачів [86]. Існує три основних типи даної атаки: Reflected XSS, Stored XSS, DOM XSS. Reflected XSS виникає коли сервер включає неперевірені та неекрановані данні користувача, як частину виведення html. Зазвичай виконання трапляється за допомогою незахищених параметрів, тому для її реалізації потенційна жертва повинна перейти за посиланням для відпрацювання корисного навантаження.

Stored XSS вважається більш небезпеченим, через те що він зберігається безпосередньо на самому сервері. Наприклад скрипт може відпрацювати коли адмін або клієнт перейдуть на сторінку з коментарями в якій є вже згенероване корисне навантаження javascript. Для реалізації даного типу атаки, комунікація з клієнтом не обов`язкова. Останній тип DOM XSS при якому фреймворки JavaScript, односторінкові програми та API, які динамічно включають контрольовані зловмисниками дані на сторінку, уразливі до DOM XSS [87]. В ідеалі програма не надсилала б контрольовані зловмисником дані до небезпечних API JavaScript. Типові атаки XSS включають крадіжку сеансу, захоплення облікового запису, обхід MFA, заміну або пошкодження вузла DOM (наприклад, панелі входу трояна), атаки на браузер користувача, такі як завантаження зловмисного програмного забезпечення, логування ключів та інші атаки на стороні клієнта [88].

1.5 Firewall, як система протидії кібер атакам.

Зі зростанням атак на веб застосунки, зросла й потреба у тестуванні елементів безпеки для протидії нападів зі сторони хакерів. Одним з найпоширеніших явищ – це використання firewall. Firewall – це програмне забезпечення, яке являє собою сукупність автоматизованих захисних фільтрів, так названий захисний екран, який усуває атаки спрямовані на веб-додатки. Він фільтрує трафік, який надходить безпосередньо від клієнту до серверу, автоматично відрізняє шкідливий трафік від безпечного, але й таки способи захисту не можна назвати повністю універсальними. Як показала практика використання firewall, неможливо повністю позбавитися від шкідливого трафіку, але можливо мінімізувати використання найбільш розповсюджених та відомих методів атак на веб-додатки. На рисунку 1.7 зображена мінімалістична архітектура firewall [88].

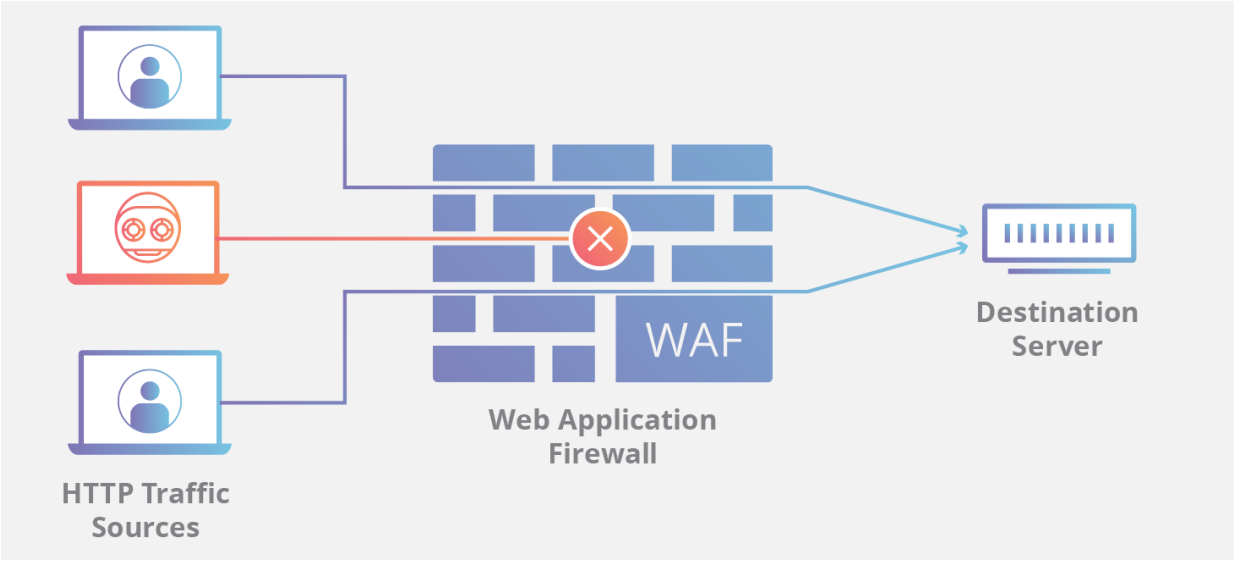

Рисунок 1.7 Архітектура firewall

Великої популярності набуло використання хмарних технологій, які поставляються разом із системами захисту від атак типу DOS, DDOS та інших типів атак на мережу. Такі компанії як Microsoft, Amazon та Google поставляють велику кількість сервісів для підтримки веб-додатку у захищеному стані. Наприклад AWS Firewall Manager пропонує клієнту потужний сервіс з налаштування фільтрів між мережевого екрану. Захист веб-додатку повинен відбуватися від розробки пз до його кінцевого розгортання у production. Для забезпечення захисту веб ресурсу існує велика кількість методів та сервісів: SAST, DAST, Source Code Review, IAST [10].

1.6 Формування задач для дипломної роботи

Проаналізувавши загальну архітектуру веб застосунків та можливі загрози, які можуть вплинути на безпеку веб застосунків, можна зформувати задачи дипломної роботи, задля вирішення проблеми ідентифікації вразилвостей у застоснуках:

- Виконати аналіз рішень щодо виявлення та ідентифікації вразливостей у веб застосунках
- Виконати аналіз архітектур веб застосунків
- Виконати аналіз методів логування подій кібербезпеки у веб застосунках

- Виконати синтез моделі виявлення і ідентифікації вразливостей у веб застосунках
- Виконати аналіз адекватності запропонованої моделі

Висновки за розділом 1

Архітектура веб застосунків має велику кількість компонентів, тому для того щоб ідентифікувати вразливість, потрібно належним чином визначити можливі загрози для застосунку. Для цього можна використати існуючу модель ідентифікації загроз Threat Modelling, але дана модель вимагає мануального втручання з боку інженера, тому задля покращення данної моделі, можна покращити процес ідентифікації вразливостей, використовуючи автоматизацію у вигляді роботи штучного інтелекту, який буде основуватися на всіх ендпоінтах веб застосунку, тобто використовувати інформацію про події, що відбуваються у веб застосунках. Тобто задачею є моніторінг всіх елементів архітектури за допомогою інструментів моніторінгу, збираючи інформацію в лог-файли. На основі цих лог-файлів штучний інтелект може зазначити вероґідний тип атаки, яка відбулася у ключовій системі, що зменшить час реагування на інциденти та знизить ризики невірної ідентифікації вразливості.

РОЗДІЛ 2
ЗБІР ІНФОРМАЦІЇ ПРО ПОДІЇ У ВСІХ ЕЛЕМЕНТАХ АРХІТЕКТУРИ ВЕБ ЗАСТОСУНКІВ

Використання функціх логування подій у веб сервері, допомогає інженерам з кібербезпеки та системним адміністраторам краще розуміти, що відбувається в застосунку, так як мають достатньо інформації, яка формує лог-файли, щоб описати конкретні дії з боку клієнтів по відношенню до веб серверу. Саме тому належне використання систем логування може надати можливість використати ці данні для подальшого аналізу, та на їх основі сформувати dataset, який у подальшому може бути використаний нейронною мережею для вирішення проблеми з автоматизацією при ідентифікації вразливостей у веб застоснуках [89].

2.1 Логування подій у веб застосунку

Логування подій у всіх елементах веб серверу є важливим аспектом забезпечення безпеки та надійності роботи веб застосунку. Існує кілька способів логування подій, які можна використовувати у всіх елементах веб серверу.

1.Логування подій на рівні операційної системи: Операційна система може надавати інформацію про події, які сталися на рівні ядра системи або вище. Це можуть бути події такі як запуск або зупинка служби, вхід або вихід з системи, зміни в реєстрі або налаштуваннях системи. Цю інформацію можна логувати в спеціальний журнал подій (Event Log), який можна переглянути і аналізувати пізніше [90].

2.Логування подій на рівні веб-сервера: Веб-сервер може логувати події, які сталися на рівні HTTP-запитів та відповідей. Це можуть бути події такі як запит на сторінку, передача даних на сервер, відправка відповіді клієнту, тощо. Лог-файли можуть бути збережені на сервері або відправлені на центральний сервер логування.

3.Логування подій на рівні застосунку: Веб застосунок може логувати події, які сталися на рівні додатку. Це можуть бути події такі як виконання запиту до бази

даних, збереження даних, зміна стану користувача, тощо. Лог-файли можуть бути збережені в базі даних або відправлені на центральний сервер логування.

4.Логування подій на рівні клієнтської сторони: Клієнтська сторона веб застосунку також може логувати події, які сталися в процесі взаємодії користувача зі сторінкою. Це можуть бути події такі як клік на кнопку, введення даних у форму, перехід на іншу сторінку, тощо. Лог-файли можуть бути збережені на клієнтській машині або відправлені на центральний сервер логування [2].

Для логування подій у всіх елементах веб серверу можна використовувати спеціальні бібліотеки та інструменти, які дозволяють легко налаштувати та керувати логуванням. Також важливо розробляти систему логування з урахуванням безпеки, щоб забезпечити конфіденційність та цілісність лог-файлів, а також захист від можливих атак на систему логування [92].

2.2 Логування подій у веб застосунку на рівні операційної системи

Логування на рівні операційної системи можна налаштувати для реєстрації подій, які відбуваються на рівні операційної системи, такі як створення нового користувача, вхід до системи, запуск служб або програм, помилки системи, тощо. Для налаштування логування на рівні операційної системи можна виконати наступі кроки: Відкрити діалогове вікно налаштувань логування. Це може бути виконано залежно від операційної системи, яку використовується. Наприклад, в Windows це може бути зроблено через Панель управління -> Журнали подій, а в Linux можна скористатися наступною командою в терміналі: sudo nano /etc/rsyslog.conf. Після цього обрати тип подій, які потрібно логувати. Операційні системи зазвичай пропонують різні рівні логування, такі як інформаційний, попередження та помилки. Нам потрібно вибрати ті типи подій, які є важливими для системи. Як тільки було обрано рівні логуваняя можна встановити місце зберігання лог-файлів. Це може бути директорія на жорсткому диску або мережева папка. Важливо встановити місце зберігання таким чином, щоб уникнути заповнення диска та забезпечити безпеку та захист від несанкціонованого доступу [93]. Для цього потрібно вірно налаштувати параметри

логування. Це може включати формат лог-файлів, рівень деталізації логування, терміни зберігання лог-файлів та інші параметри. Також повинно перевірятись налаштування логування, переконуючись, що лог-файли зберігаються в правильному місці, з правильним форматом та деталізацією, також перевірте, що вони не заповнюють диск і що вони доступні для перегляду адміністратором системи. Для перевірки лог-файлів можна використовувати різні інструменти, наприклад, Event Viewer у Windows або команду grep у Linux.

Встановити механізм ротації лог-файлів є обов`язковим кроком. Якщо лог-файли стають надто великими, вони можуть спричинити перевантаження системи та заповнення диску. Для цього можна налаштувати механізм ротації лог-файлів, який дозволить періодично створювати нові файли та архівувати старі. Не потрібно забувати про моніторинг лог-файлів. Для того, щоб швидко виявляти проблеми в системі, необхідно налаштувати моніторинг лог-файлів. Це може бути зроблено за допомогою спеціальних інструментів, наприклад, Splunk або Nagios, які зможуть автоматично виявляти та повідомляти про проблеми в системі на основі даних з лог-файлів. Налаштування логування на рівні операційної системи дозволить нам відстежувати події, які відбуваються на рівні операційної системи, та швидко виявляти проблеми в системі. Це є важливим елементом забезпечення безпеки та стабільності веб застосунків.

2.3 Логування подій у веб застосунку на рівні веб серверу

Налаштування логування на рівні веб-сервера є важливою частиною забезпечення безпеки та стабільності веб застосунків. Хочу описати деякі кроки, які можна виконати для налаштування логування на рівні веб-сервера.

Перш за все необхідно обрати формат лог-файлів. Формат лог-файлів може бути різним, в залежності від веб-сервера, який використовується. Найпоширеніші формати - це Common Log Format (CLF) та Combined Log Format (CLF), які містять інформацію про запити клієнтів, статус відповіді сервера, розмір файлу тощо. Також можна налаштувати запис лог-файлів. Можна визначити, які події слід логувати в лог-

файлах веб-сервера. Наприклад, можна включити логування запитів клієнтів, відповідей сервера, помилок, затримок відповіді тощо. Також можна встановити рівень деталізації лог-файлів, щоб вони не заповнювали диск і їх можна було легко переглядати. Після цього потрібно встановити розмір лог-файлів та механізм ротації. Можна встановити максимальний розмір лог-файлів та механізм ротації, який дозволить періодично створювати нові файли та архівувати старі, щоб уникнути перевантаження системи та заповнення диску [93].

Одним з головних пунктів є налаштування рівня доступу до лог-файлів. Ми можемо встановити права доступу до лог-файлів, щоб забезпечити безпеку даних та запобігти несанкціонованому доступу до лог-файлів. Включіть логування SSL-запитів, якщо вони використовуються. Якщо ми використовуємо SSL-з'єднання для шифрування трафіку на вашому веб-сервері, то слід включити логування SSL-запитів. Це дозволить відслідковувати з'єднання з шифруванням, а також забезпечити додатковий захист для вашого сервера. Використовувати інструменти аналізу лог-файлів теж необхідно. Інструменти аналізу лог-файлів можуть допомогти нам зрозуміти, як використовується ваш веб-сервер, які запити виконуються найбільше, які сторінки найбільше відвідуються тощо. Ці дані можуть допомогти покращити продуктивність веб-сервера та додатку, а також зрозуміти, які проблеми можуть виникнути у майбутньому.

Забезпечення резервного копіювання лог-файлів є обов’язковим кроком у підтриманні цілісності наших даних. Резервне копіювання лог-файлів може допомогти зберегти дані, які містяться в цих файлах у разі їх втрати або пошкодження. Ми можемо скопіювати лог-файли на інший сервер або використовувати інші механізми резервного копіювання, які підходять для нашого середовища. Встановлення моніторингу лог-файлів. Ми можемо використовувати моніторинг лог-файлів для виявлення помилок та проблем, які можуть виникнути на вашому веб-сервері. Моніторинг може бути автоматизованим, або можна ж вручну переглядати лог-файли з регулярністю, що не бажано [94].

Узагальнюючи, налаштування логування на рівні веб-сервера може бути важливим для забезпечення безпеки та стабільності вашого веб застосунку.

Налаштування лог-файлів, ротація, доступ до лог-файлів, аналіз та моніторинг можуть допомогти вам отримати доступ до цінних даних про веб-сервер, таких як запити користувачів, помилки, відповіді сервера та інші важливі інформаційні події. Налаштування логування на рівні веб-сервера може здатися складним процесом, але воно дуже важливе для забезпечення безпеки та стабільності веб застосунку. Пам'ятайте, що краще запобігти проблемам, ніж потім вирішувати їх на запитання користувачів або в результаті атак на ваш веб-сервер [93].

2.4 Логування подій у веб застосунку на рівні веб застосунку

Налаштування логування на рівні веб застосунку дозволяє контролювати те, що відбувається в самому додатку. Це може допомогти відстежувати дії користувачів, виявляти помилки та вдосконалювати функціонал програми. Для налаштування логування на рівні веб застосунку можна використовувати бібліотеки логування, такі як Log4j, Logback або NLog, які допомагають управляти журналами подій. Ось деякі кроки, які потрібно виконати для налаштування логування на рівні веб застосунку:
Встановити бібліотеку логування: Почати з встановлення бібліотеки логування нашого вибору. Зазвичай це залежить від мови програмування та фреймворка. Наприклад, для Java-додатків можна використовувати бібліотеки, такі як Log4j або Logback.Налаштуйте журнал подій: Після встановлення бібліотеки логування вам потрібно налаштувати журнал подій. Ви можете налаштувати, які типи подій слід логувати, та рівень журналування, такий як DEBUG, INFO, WARN або ERROR.

Використовуйте логер: Після налаштування журналування використовуйте логер для записування подій у журнал. Ви можете використовувати логер у вашому коді, щоб записувати події, які відбуваються в вашому додатку. Аналізуйте журнали подій: Після того, як ви записали деякі події в журнал, ви можете проаналізувати їх, щоб зрозуміти, що відбувається в вашому додатку. Налаштуйте розширене логування: Деякі бібліотеки логування дозволяють налаштовувати розширене логування, таке як логування запитів HTTP або SQL-запитів. Це допомагає збирати більше інформації про взаємодію вашого додатку з зовнішніми системами.

Використовуйте аналітику: Ви можете використовувати аналітику, щоб аналізувати дані, зібрані з логів вашого додатку. Наприклад, ви можете використовувати аналітику, щоб знайти найбільш популярні функції в вашому додатку або виявити помилки, які впливають на багато користувачів. Налаштуйте моніторинг: Для ефективного виявлення проблем в вашому додатку в реальному часі використовуйте моніторинг логів. Це може допомогти вам відстежувати помилки, що виникають в реальному часі, і відповідно реагувати на них. Захищайте дані: Під час логування подій у вашому додатку важливо захистити дані від несанкціонованого доступу. Переконайтеся, що журнали подій зберігаються в захищеному місці і доступ до них мають тільки авторизовані користувачі. Перевіряйте журнали регулярно: Перевіряйте журнали подій регулярно, щоб переконатися, що ваш додаток працює правильно. Перевірка журналів може допомогти виявити проблеми, які не були помічені раніше. Підтримуйте логування: Підтримуйте логування у вашому додатку, оновлюйте бібліотеки логування, перевіряйте правильність налаштування журналування та встановлюйте нові версії програмного забезпечення, якщо це потрібно. Також, регулярно аналізуйте дані, зібрані з журналів, щоб виявляти тенденції та покращувати ефективність вашого додатку. Документуйте журнали подій: Під час розробки вашого додатку документуйте журнали подій. Описуйте, яку інформацію збирається, які поля включає кожен запис журналу та які формати використовуються для збереження журналів. Це допоможе іншим розробникам швидко зрозуміти, як працює ваш додаток та які інформаційні ресурси можна використовувати для розв'язання проблем. Дотримуйтесь правил безпеки: Під час логування подій важливо дотримуватися правил безпеки. Наприклад, не зберігайте паролі або конфіденційну інформацію у журналах подій. Також, переконайтеся, що доступ до журналів подій обмежений ізоляцією серверів та захистом мережі. Розгляньте використання спеціальних інструментів: Існує багато спеціальних інструментів для збору та аналізу логів додатків. Наприклад, ELK (Elasticsearch, Logstash, Kibana) - це відкрите рішення для збору, аналізу та візуалізації логів. Інші інструменти можуть надавати розширені можливості аналізу даних, такі як машинне навчання або розумний аналіз. Налаштуйте систему оповіщення: Налаштуйте систему оповіщення для швидкої

реакції на критичні проблеми, виявлені в логах. Наприклад, ви можете налаштувати систему, щоб відправляти сповіщення на електронну пошту адміністратора або відправляти повідомлення на службу технічної підтримки. Це допоможе швидко виявляти та вирішувати проблеми у вашому веб-додатку. Тестуйте систему логування: Перевірте, що ваша система логування працює правильно. Проводьте тестування різних сценаріїв, щоб переконатися, що журнали подій збираються та зберігаються правильно. Також, тестуйте систему оповіщення, щоб переконатися, що вона працює належним чином [94].

Логування подій є важливою складовою веб-додатків, які допомагають виявляти та вирішувати проблеми в режимі реального часу. Для налаштування логування на різних рівнях веб-додатка, включаючи операційну систему, веб-сервер та сам додаток, важливо дотримуватися правил безпеки та використовувати найкращі практики. Налаштування системи логування потребує певного часу та зусиль, але це дозволить забезпечити високу доступність вашого веб-додатка та підтримувати його ефективність на довгий час [93].

2.5 Логування подій у веб застосунку на рівні клієнта

Логування на клієнтській стороні (front-end) може допомогти виявляти та вирішувати проблеми, пов'язані зі збоєм на клієнтській стороні, такі як помилки JavaScript, проблеми з мережевими запитами або проблеми з рендерингом сторінок. Основні етапи налаштування логування на клієнтській стороні веб-додатка:

Виберіть бібліотеку або фреймворк: Існує безліч бібліотек та фреймворків, які допомагають логувати події на клієнтській стороні. Деякі з них вже мають вбудовану підтримку логування, наприклад, Angular, React та Vue.js. Інші бібліотеки можуть використовувати сторонні бібліотеки, такі як Log4Javascript або LogRocket.

Включіть логування: Налаштуйте вашу бібліотеку або фреймворк для включення логування. Це може вимагати налаштування параметрів та включення додаткового коду. Наприклад, у React можна використовувати console.log() для логування, або встановити спеціальну бібліотеку, таку як redux-logger.

Визначте рівень логування: Оберіть рівень логування для клієнтської сторони вашого додатка. Рівні логування можуть включати наступне: "debug", "info", "warn" та "error". Рівень логування може бути налаштований для кожної окремої події, щоб забезпечити максимальну гнучкість.

Зберігайте журнали подій: Журнали подій можна зберігати на клієнтській стороні (наприклад, у локальному сховищі), або надсилати їх на сервер для подальшого аналізу. При цьому потрбно пам'ятати про те, що обсяг даних, які будуть зберігатися, може бути досить великим, тому потрібно забезпечити оптимальність обробки та передачі даних.

Аналізуйте дані: Після збору логів важливо проаналізувати отримані дані та вирішити проблеми, які були виявлені. Для цього можна використовувати спеціальні інструменти аналізу журналів, такі як Kibana, Loggly, або Splunk.

Захистіть дані: Забезпечте захист зібраних логів, оскільки вони можуть містити конфіденційну інформацію, таку як імена користувачів та паролі. Для захисту логів можна використовувати шифрування даних, контроль доступу та інші методи безпеки.

Отже, налаштування логування на клієнтській стороні може допомогти виявляти та вирішувати проблеми, пов'язані зі збоєм на клієнтській стороні, та покращити загальну продуктивність вашого додатка [94].

2.6 Створення dataset

Логи веб-сервера містять велику кількість інформації, яка може бути використана для створення датасету для ідентифікації вразливостей. Деякі з найбільш важливих даних, які можна використати для цієї мети, включають:

- Запити до сервера: Ці дані містять інформацію про те, які запити були відправлені до сервера. Наприклад, це можуть бути запити GET, POST, PUT або DELETE.

• URL-адреси: Ці дані містять інформацію про те, які сторінки були запитані. URL-адреси можуть вказувати на конкретні сторінки, які можуть бути вразливими до атак.

• Параметри запиту: Ці дані містять інформацію про параметри запиту, такі як імена користувачів, паролі та інші дані, які можуть бути використані для атак.

• Коди стану HTTP: Ці дані містять інформацію про те, як сервер обробив запит. Коди стану можуть вказувати на помилки, які можуть бути використані для атак.

• Інформація про браузер: Ці дані містять інформацію про браузер, який був використаний для взаємодії з сервером. Ця інформація може допомогти в ідентифікації вразливостей, пов'язаних з конкретними браузерами.

Дані про користувачів: Ці дані містять інформацію про користувачів, таку як IP-адреси, імена користувачів та інші дані, які можуть бути використані для ідентифікації вразливостей.

Ці дані можуть бути використані для створення датасету, який можна використовувати для тренування нейронної мережі для ідентифікації вразливостей.
Якщо більш детально аналізувати інформацію, що подається у логах веб-серверу, то можна з більшою долею вірогідності ідентифікувати тип вразливості у веб застосунку, пропоную розглянути наступні данні:

Запити клієнтів: Дані про запити, які надійшли до сервера, можуть містити інформацію про тип запиту, параметри, адресу запиту та інші параметри, які можуть вказувати на потенційну вразливість. Наприклад, запити, які містять SQL-код в параметрах, можуть вказувати на можливу SQL-ін'єкцію.
Відповіді сервера: Інформація про відповіді сервера може також містити важливу інформацію про можливі вразливості. Наприклад, відповіді, які містять інформацію про помилку сервера, можуть вказувати на можливу уразливість, яку можна використати для злому.

Логи автентифікації: Логи автентифікації містять інформацію про спроби входу в систему. Вони можуть містити інформацію про невдалі спроби входу, які можуть

свідчити про спроби атакувати систему через вразливості в механізмах автентифікації.

Логи доступу до файлів: Логи доступу до файлів містять інформацію про те, які файли були запитані і які були відправлені клієнту. Ці дані можуть допомогти виявити вразливості, пов'язані з недостатньою авторизацією на доступ до файлів або вразливості в механізмах контролю доступу.

Логи аудиту: Логи аудиту містять інформацію про дії користувачів, які виконуються в системі. Ці дані можуть допомогти виявити потенційні вразливості, пов'язані з незвичайними діями користувачів.

Час відповіді сервера: Вимірювання часу відповіді сервера може допомогти виявити низькопродуктивні або погано налаштовані сервери, які можуть бути вразливими до атак на затримку або доS-атак.

Розмір відповіді: Розмір відповіді сервера може також бути корисним для ідентифікації вразливостей, таких як переповнення буфера або вразливість на вимушене завантаження даних, які можуть призвести до збоїв сервера або витоку інформації.

Код стану HTTP: Код стану HTTP вказує на успішність або невдачу запиту до сервера. Коди помилок (наприклад, 404 "сторінка не знайдена" або 500 "внутрішня помилка сервера") можуть вказувати на вразливості, які можуть бути використані зловмисниками для атак на веб-сайти [96].

Запити до баз даних: Логи веб-сервера можуть також містити запити до баз даних, які використовуються для зберігання та обробки даних на сервері. Ці дані можуть містити інформацію про вразливості баз даних, такі як SQL-ін'єкції, які можуть дозволити зловмисникам виконувати небезпечні запити до баз даних та отримувати невідповідну інформацію.

Додаткові заголовки запиту: Логи веб-сервера можуть містити інформацію про додаткові заголовки запиту, які надсилаються з браузера до сервера. Ці дані можуть містити інформацію про вразливості браузерів, такі як XSS-атаки або використання вразливих куків.

2.7 Аналіз лог файлів

На рисунку 2.1 можна побачити SQL запит до таблиці users, в якому використовується уразливість SQL ін'єкції. Уразливість полягає у тому, що значення параметрів запиту (username та password) не були захищені від вставки додаткових символів, які можуть змінити поведінку запиту.

```
[2022-03-15 12:04:32] [ERROR] [sqlalchemy.engine.base.Engine] SELECT * FROM
users WHERE username='admin' AND password='password' OR 1=1;
```

Рисунок 2.1 Приклад лог-файлу з фреймворку Django версії 2.0

У цьому конкретному запиті значення параметра password було встановлено як "password' OR 1=1", що змінює логіку запиту. Замість того, щоб шукати користувача з іменем "admin" та паролем "password", запит поверне будь-якого користувача, оскільки умова "1=1" завжди буде істинною.

Іншим прикладом може бути наступний рядок логів на рисунку 2.2.

```
[2022-03-15 13:23:41] [WARNING] [django.security.csrf] Forbidden (CSRF token
missing or incorrect.): /change_password/
```

Рисунок 2.2 Приклад лог-файлу з фреймворку Django версії 3.0

У цьому рядку можна побачити повідомлення про помилку CSRF токену на сторінці зміни пароля. CSRF або Cross-Site Request Forgery - це тип атаки, коли зловмисник може виконати певні дії в ім'я користувача, використовуючи вразливість веб застосунку, який дозволяє виконувати запити без підтвердження дій користувача [72].

У цьому конкретному випадку, можливо, що на сторінці зміни пароля не було додано CSRF токену, який повинен бути використаний для підтвердження дій користувача. Це може відкрити двері для атак, які можуть змінити пароль користувача без його дозволу.

Ці логи можна також використовувати для створення датасету, який буде використовуватися нейронною мережею для виявлення подібних вразливостей в майбутньому. Ще один приклад можна побачити наступний рядок логів на рисунку 2.3:

```
[2022-03-15 13:45:20] [ERROR] [django.db.backends] (1054, "Unknown column 'email' in 'field list'")
```

Рисунок 2.3 Приклад лог-файлу з фреймворку Django версії 3.3

У цьому рядку можна побачити повідомлення про помилку в запиті до бази даних, де згадується стовпчик email, який не існує в таблиці. Це може бути викликано написанням SQL запиту з помилкою, де назва стовпчика неправильна або таблиця була змінена, і стовпець більше не існує. Це може відкрити двері для SQL-ін'єкцій, коли зловмисник може використати цю помилку для внесення шкідливого коду у запит до бази даних. Ці логи можна також використовувати для створення датасету, який буде використовуватися нейронною мережею для виявлення SQL-ін'єкцій в майбутньому. Шляхом аналізу таких помилок можна створити систему, яка буде виявляти підозрілі запити і запобігати можливим атакам на веб застосунок [96].

Останнім прикладом у аналізі логів буде послідовні запити до сервера які містять потенційні sql запити у GET параметрах. Для візуалізіції буду наводити приклад з використанням лог файлу від веб серверу Apache. На рисунку 2.4 зазначиний приклад використання мови програмування PHP7, як основної мови додатку [95]. Взагалі структура логів не дуже сильно міняється з виходом нової версії будь-якого фреймворку, саме тому по лог файлах з легкістю можна автоматизовано

проводити аналіз будь-яких подій. В нашому випадку використання моделі нейронної мережі для ідентифікації типу веб вразливості.

```
2022-03-28 12:14:43 127.0.0.1 POST /login.php - 443 - 127.0.0.1
Mozilla/5.0+(Windows+NT+10.0;+Win64;+x64;+rv:90.0)+Gecko/20100101+Firefox/90.
0 - 200 0 0 150
2022-03-28 12:14:46 127.0.0.1 GET /user.php?id=3 - 443 - 127.0.0.1
Mozilla/5.0+(Windows+NT+10.0;+Win64;+x64;+rv:90.0)+Gecko/20100101+Firefox/90.
0 - 200 0 0 85
2022-03-28 12:14:47 127.0.0.1 GET /user.php?id=1+or+1=1 - 443 - 127.0.0.1
Mozilla/5.0+(Windows+NT+10.0;+Win64;+x64;+rv:90.0)+Gecko/20100101+Firefox/90.
0 - 200 0 0 95
2022-03-28 12:14:49 127.0.0.1 GET /user.php?id=3+or+1=1 - 443 - 127.0.0.1
Mozilla/5.0+(Windows+NT+10.0;+Win64;+x64;+rv:90.0)+Gecko/20100101+Firefox/90.
0 - 200 0 0 95
2022-03-28 12:14:50 127.0.0.1 GET /user.php?id=1+union+select+1,2,3 - 443 - 127.0.0.1
Mozilla/5.0+(Windows+NT+10.0;+Win64;+x64;+rv:90.0)+
```

Рисунок 2.4 Приклад лог-файлу веб серверу Apache мови програмування PHP7

У логах можна помітити певну вразливість, яка проявляється у несанкціонованому доступі до даних користувачів.

У логах видно, що було виконано POST-запит до файлу /login.php, що означає спробу входу у систему. Далі, були виконані GET-запити до файлу /user.php з різними параметрами id, що означає запит інформації користувача з зазначеним id.

Однак, на останньому рядку логів був виконаний GET-запит з параметром ID, що містить SQL-ін'єкцію, що може бути використане для отримання певної конфіденційної інформації про користувачів [46].

Таким чином, логи вказують на можливу вразливість у системі та необхідність вжиття заходів для її виправлення [95].

Висновки за розділом 2

Логування подій у всіх елементах веб серверу є важливим аспектом забезпечення безпеки та надійності роботи веб застосунку. Існує кілька способів логування подій, які можна використовувати у всіх елементах веб серверу: логування подій на рівні операційної системи, логування подій на рівні веб-сервера, логування подій на рівні застосунку, логування подій на рівні клієнтської сторони. Логи веб-сервера містять велику кількість інформації, яка може бути використана для створення датасету для ідентифікації вразливостей. Деякі з найбільш важливих даних, які можна використати для цієї мети, включають: запити до сервера: Ці дані містять інформацію про те, які запити були відправлені до сервера. Наприклад, це можуть бути запити GET, POST, PUT або DELETE. URL-адреси: ці дані містять інформацію про те, які сторінки були запитані. URL-адреси можуть вказувати на конкретні сторінки, які можуть бути вразливими до атак. Параметри запиту: ці дані містять інформацію про параметри запиту, такі як імена користувачів, паролі та інші дані, які можуть бути використані для атак і тд. Отже для компанування датасету, який буде використаний нейронною мережею для навчання можна використати логи веб-серверу. Вони містять багато корисної інформації. Їх можна представити у певній класифікації завдяки тому, що вони мають одну й ту саму структуру будування.

РОЗДІЛ 3
РОЗРОБКА МОДЕЛІ ІДЕНТИФІКАЦІЇ ВРАЗЛИВОСТЕЙ У ВЕБ ЗАСТОСУНКАХ

3.1 Типи нейроних мереж

Нейронні мережі - це комп'ютерні системи, що навчаються, здатні до розв'язання різноманітних завдань, включаючи класифікацію, розпізнавання образів, прогнозування тощо. Основною їх ідеєю є математичне моделювання роботи людського мозку [97].

Основні типи нейронних мереж:

- Прямі нейронні мережі (Feedforward Neural Networks) - найпоширеніший тип нейронних мереж, що складаються зі шарів нейронів, які передають інформацію від вхідного до вихідного шару.
- Рекурентні нейронні мережі (Recurrent Neural Networks) - мережі, що мають зв'язки між нейронами у вигляді циклів, що дозволяє моделювати послідовності даних, такі як мовлення та текст.
- Згорткові нейронні мережі (Convolutional Neural Networks) - спеціалізований тип нейронних мереж для обробки зображень, що використовують згортку та пулінг для виявлення рис та форм.
- Автокодери (Autoencoders) - мережі, які навчаються зменшувати розмірність вхідних даних з метою виявлення складових та зменшення шуму.
- Супутні навчання (Transfer Learning) - використання навчення попередньої моделі для розв'язання нових задач.

• Глибокі нейронні мережі (Deep Neural Networks) - мережі з великою кількістю шарів, що дає змогу отримати більш складні та точні результати, але при цьому потребує більшої кількості обчислювальних ресурсів та даних для навчання.

• Генеративні застосування (Generative Adversarial Networks) - мережі, які можуть генерувати нові дані на основі вхідних даних та покращувати їх якість шляхом змагання двох нейронних мереж: генеративної та дискримінативної.

• Самоорганізуючіся карти Кохонена (Self-Organizing Maps) - мережі, що використовуються для візуалізації та аналізу даних з високою розмірністю.

• Багатоагентні системи (Multi-Agent Systems) - мережі, що складаються з багатьох агентів, кожен з яких має свої власні параметри та поведінку.

• Еволюційні нейронні мережі (Evolutionary Neural Networks) - мережі, що використовуються для автоматичного підбору оптимальної архітектури та гіперпараметрів нейронних мереж [107].

Кожен тип нейронної мережі має свої особливості та призначення, тому важливо вибрати правильний тип для конкретної задачі. Також важливим є вибір оптимальних гіперпараметрів та архітектури мережі для досягнення найкращих результатів [98].

3.2 Рекурентні нейронні мережі

Рекурентні нейронні мережі є одним з найбільш поширених типів нейронних мереж, і вони зазвичай використовуються для розв'язання задач класифікації та регресії. Задачі класифікації полягають у призначенні кожному елементу вхідних даних (наприклад, зображенню) певного класу, який визначається за певними ознаками. Рекурентні нейронні мережі добре справляються з цією задачею завдяки своїй здатності до розпізнавання складних взаємозв'язків між даними та їх класами. Задачі регресії полягають у прогнозуванні числових значень на основі даних. Прямі нейронні мережі також є ефективним інструментом для розв'язання цих задач,

оскільки вони можуть навчатися з розширеними даними та виробляти точні прогнози, що базуються на складних взаємозв'язках між даними [100].

Отже, рекурентні нейронні мережі є ефективним інструментом для розв'язання задач класифікації та регресії, особливо у випадках, коли маються великі обсяги даних та складні взаємозв'язки між ними.

Виходячи з цього для моделювання власної моделі для ідентифікації вразливостей у веб застосунках краще використати саме рекурентний принцип на основі дерев рішень, який якраз підійде для класифікації типу атаки, що відбулася у застосунку.

3.3 Дерева рішень

Дерева рішень - це метод машинного навчання, який використовується для прийняття рішень в умовах невизначеності. Цей метод дозволяє створювати моделі, які аналізують вхідні дані та приймають рішення на основі набору правил, що визначаються в процесі навчання. Дерева рішень складаються з вузлів та гілок. Кожен вузол представляє певну характеристику або ознаку, що характеризує вхідні дані, а кожна гілка представляє варіант рішення, яке може бути прийняте на основі цієї ознаки. Кожен вузол має декілька гілок, які ведуть до інших вузлів, або до кінцевих листків дерева, які представляють варіанти рішень [99].

Для створення моделі дерева рішень необхідно навчити алгоритм на основі набору прикладів з відомими правильними рішеннями. Алгоритм використовує ці дані для побудови дерева, яке може бути використане для прийняття рішень на нових вхідних даних.Дерева рішень мають декілька переваг, таких як:

- Легко зрозумілі та інтерпретовані: дерева рішень можна легко візуалізувати, що дозволяє зрозуміти логіку прийняття рішень.
- Можуть використовуватись для роботи з багатьма типами даних: дерева рішень можуть бути використані для роботи з різними типами даних, включаючи текст, зображення та звук.

• Мають високу точність: дерева рішень можуть давати високу точність у багатьох випадках, зокрема, якщо модель має достатньо велику кількість прикладів для навчання.

Дерева рішень також мають деякі недоліки:

• Схильність до перенавчання: дерева рішень можуть стати надто складними та перенавчатись на навчальних даних, що може призвести до низької точності при роботі з новими даними.

• Низька стійкість до зміни даних: дерева рішень можуть бути дуже чутливими до навчальних даних та можуть виявлятись не ефективними в тих випадках, коли дані змінюються з часом.

• Обмежена здатність до роботи з багатовимірними даними: дерева рішень можуть бути обмеженими у своїй здатності до обробки багатовимірних даних.

Дерева рішень використовуються в різних областях, таких як фінанси, медицина, бізнес та інші. Вони можуть використовуватись для вирішення різних задач, таких як класифікація даних, прогнозування тенденцій, оцінка ризиків та інші. Застосування дерев рішень є досить широким і залежить від конкретного завдання, що потрібно вирішити [106].

3.4 Рандомний ліс

Алгоритм випадкового лісу можна розглядати як комбінацію двох підходів: багато дерев рішень та бутстрепової підвибірки. Багато дерев рішень:

Випадковий ліс будує декілька (зазвичай, декілька сотень) різних дерев рішень. Кожне дерево вирішує задачу класифікації або регресії зі свого власного підмножини навчальних даних. При цьому, щоб забезпечити варіативність моделей, кожне дерево будується на випадково вибраній підмножині ознак [105].

Бутстрепова підвибірка

Щоб забезпечити різноманітність моделей та зменшити їх взаємозв'язок, для навчання кожного дерева випадково вибирається підмножина навчальних даних з повторенням. Це називається бутстреповою підвибіркою. Кожна підвибірка містить тільки деяку частину з усієї множини навчальних даних, що дозволяє кожному дереву бачити трохи інші дані.Таким чином, випадковий ліс використовує багато дерев рішень, кожне з яких навчено на випадково вибраному підмножині ознак та бутстреповій підвибірці даних. Інтуїтивно, такий підхід дозволяє побудувати більш точну та стійку до перенавчання модель.

Математично обгрунтувати випадковий ліс складніше, але можна використати концепції теорії ймовірності та статистики, щоб пояснити, чому він працює добре. Зокрема, можна показати, що випадковий ліс має низьку в'язкість (low variance)

Низька в'язкість (low variance) означає, що випадковий ліс дозволяє уникнути перенавчання. Якщо модель занадто добре вивчає навчальні дані, вона може бути надто специфічною та невірно передбачати нові дані. З іншого боку, якщо модель занадто проста, вона може бути недостатньою для вирішення складних задач. Випадковий ліс дозволяє балансувати між цими двома екстремами, будуючи багато різних дерев та об'єднуючи їх прогнози.

Висока здатність до узагальнення (high generalization ability) означає, що випадковий ліс добре передбачає нові дані. Це досягається завдяки бутстреповій підвибірці, що дозволяє кожному дереву бачити трохи інші дані, та випадково вибраній підмножині ознак, що забезпечує варіативність моделей. Можна показати, що випадковий ліс має нижню межу точності (lower bound on accuracy), тобто в середньому його точність не буде гіршою, ніж деяка теоретично можлива мінімальна

точність. Це дає нам впевненість, що випадковий ліс не буде надто поганим на будь-якій задачі класифікації або регресії [106].

Отже, випадковий ліс - це потужний та ефективний алгоритм машинного навчання, який забезпечує низьку в'язкість, високу здатність до узагальнення та має нижню межу точності. Це дозволяє йому добре працювати на багатьох різних задачах та даних, що робить його одним з найпопулярніших алгоритмів машинного навчання.

3.5 Створення датасету для навчання нейрнної мережі

Створення dataset для прямої нейронної мережі передбачає підготовку даних, які будуть використовуватись для навчання, перевірки та тестування мережі. Для цього можна виконати наступні кроки:

- Зібрати та обробити дані: визначити, які дані потрібно використовувати для тренування мережі та зібрати їх. Потім обробити дані, включаючи їх очищення та попередню обробку, якщо необхідно.
- Розділити дані на тренувальну, перевірочну та тестову вибірки: тренувальна вибірка використовується для навчання мережі, перевірочна - для налаштування гіпер параметрів та валідації моделі, а тестова - для оцінки точності моделі на нових даних.
- Кодування даних: зазвичай необхідно закодувати дані у числовому форматі, щоб можна було використовувати їх для навчання мережі. Це можна зробити за допомогою різних методів кодування, таких як one-hot encoding.
- Нормалізація даних: нормалізація даних допомагає уникнути проблем зі збільшенням значення одних особливостей та зменшенням інших, що може призвести до поганої роботи мережі. Нормалізацію можна виконати за допомогою різних методів, наприклад, зміни діапазону або стандартизації [102].
- Створення dataset: після того, як дані були оброблені та нормалізовані, можна створити dataset. Dataset можна створити за допомогою бібліотек для роботи з

даними, таких як Pandas, та зберегти його у форматі, що відповідає вимогам конкретної бібліотеки.

• Крім того, Dataset можна побудувати з використанням готових наборів даних, які доступні онлайн.

З

Отже, створення dataset для прямої нейронної мережі може зайняти досить багато часу і зусиль, але якщо dataset побудований якісно, то це може дати дуже хороші результати при навчанні моделі. Важливо також пам'ятати про збалансованість dataset і його розмір, щоб уникнути перенавчання або недостатнього навчання моделі [95].

Виходячи, з усього вище сказано приступимо до створення dataset за допомогою мови програмування, на основі логів веб-серверу [100]. Мова програмування Python має величезну кількість фреймворків та готових бібліотек для створення нейронних мереж, саме тому будемо використовувати її. Також для створення dataset нам необхідно створити файл у форматі json, для легшого перевіряння логів. Підготовлений файл буде виглядати наступним чином:


```
[
  {"data": "2022-03-28 12:14:43 127.0.0.1 POST /login.php - 443 - 127.0.0.1 Mozilla/5.0+(Windows+NT+10.0;+Win64;+x64;+rv:90.0)+Gecko/20100101+Firefox/90.0 - 200 0 0 150"},
  {"data": "2022-03-28 12:14:44 127.0.0.1 GET /user.php?id=1 - 443 - 127.0.0.1 Mozilla/5.0+(Windows+NT+10.0;+Win64;+x64;+rv:90.0)+Gecko/20100101+Firefox/90.0 - 200 0 0 85"},
  {"data": "2022-03-28 12:14:45 127.0.0.1 GET /user.php?id=2 - 443 - 127.0.0.1 Mozilla/5.0+(Windows+NT+10.0;+Win64;+x64;+rv:90.0)+Gecko/20100101+Firefox/90.0 - 200 0 0 85"},
  {"data": "2022-03-28 12:14:46 127.0.0.1 GET /user.php?id=3 - 443 - 127.0.0.1 Mozilla/5.0+(Windows+NT+10.0;+Win64;+x64;+rv:90.0)+Gecko/20100101+Firefox/90.0 - 200 0 0 85"},
  {"data": "2022-03-28 12:14:47 127.0.0.1 GET /user.php?id=1+or+1=1 - 443 - 127.0.0.1 Mozilla/5.0+(Windows+NT+10.0;+Win64;+x64;+rv:90.0)+Gecko/20100101+Firefox/90.0 - 200 0 0 95"},
  {"data": "2022-03-28 12:14:48 127.0.0.1 GET /user.php?id=2+or+1=1 - 443 - 127.0.0.1 Mozilla/5.0+(Windows+NT+10.0;+Win64;+x64;+rv:90.0)+Gecko/20100101+Firefox/90.0 - 200 0 0 95"},
  {"data": "2022-03-28 12:14:49 127.0.0.1 GET /user.php?id=3+or+1=1 - 443 - 127.0.0.1 Mozilla/5.0+(Windows+NT+10.0;+Win64;+x64;+rv:90.0)+Gecko/20100101+Firefox/90.0 - 200 0 0 95"},
  {"data": "2022-03-28 12:14:50 127.0.0.1 GET /user.php?id=1+union+select+1,2,3 - 443 - 127.0.0.1 Mozilla/5.0+(Windows+NT+10.0;+Win64;+x64;+rv:90.0)+Gecko/20100101+Firefox/90.0 - 200 0 0 85"},
  {"data": "2022-03-28 12:14:46 127.0.0.1 GET /user.php?id=<?php+phpinfo();+?> - 443 - 127.0.0.1 Mozilla/5.0+(Windows+NT+10.0;+Win64;+x64;+rv:90.0)+Gecko/20100101+Firefox/90.0 - 200 0 0 85"}
]
```


Рисунок 3.1 Підготовка лог-файлу для створення dataset

На Рисунку 3.1 наведені 9 рядків лог-файлу, які представляють собою приклади атак, або загалних HTTP запитів. Для компанування json файлу будемо використовувати data ключ для кожного з логів. Взагалі для цього можна було використати мову програмування, але так як файл не досить великий, його можна відтворити мануально. Логи, які наведені в цьому прикладі, згенеровані штучно.

Даний тип використовується на веб-серверах типу Apache, і як бачно по розмітці сервер використовує мову програмування php.

3.6 Реалізація моделі нейронної мережі

Для реалізації нейронної мережі зпочатку імпортуємо необхідні бібліотеки, які мають вже реалізовані алгоритми з навчання в задачах класифікації. Хочу зазначити ще раз, що в цьому випадку для навчання моделі буде використовуватися мережа рандомних лісів, тому я імпортую клас RandomForestClassifier бібліотеки sklearn. Sklearn - є однією з найбільш популярних бібліотек для машинного навчання в Python. Вона надає широкий набір інструментів для розв'язання завдань класифікації, регресії, кластеризації та інших задач машинного навчання.

```
import json
from sklearn.ensemble import RandomForestClassifier
```

Рисунок 3.2 Імпортування класу для навчання моделі

Як зазначено на рисунку 3.2, ми не тільки імпортуємо модель для вирішення задачи класифікації а також імпортуємо внутрішню бібліотеку json, яка буде використовуватися для парсингу датасету і тестових логів.

Після імпортування необхідно зчитати данні з файлу, які наведені на рисунку 3.1. Для цього ми використовую невелику функцію яка буде відповідати за відкриття файлу та завантаження даних. За допомогою конструкції на рисунку 3.3 створюю функцію [101].

```
5   def read_data() -> list[dict]:
6       with open("data.json", 'r') as f:
7           data = json.load(f)
8
9       return data
```

Рисунок 3.3 Створення функції для зчитування даних з json файлу

На рядку 6, з коду наведеного на рисунку 3.3, завантажуємо файл з назвою data.json, який містить логи веб-серверу на рисунку 3.1. На рядку 7 створюємо змінну data, яка за допомогою функції load бібліотеки json приймає на вхід об’єкт файлу f. На рядку 9 повертаємо змінну *data*, за допомогою службового слова *return.*

Після створення функції для зчитування json файлів, починаємо реалізацію функції, яка буде перевіряти данні які повертає read_data функція. Для цього створюємо функцію parse_data.

```
12  def parse_data(data: list[dict]) -> list[list]:
13      parsed_data = []
14
15      for i in range(len(data)):
16          parsed_data.append([])
17
18          parsed_data[i] = data[i]['data'].split(' ')
19          parsed_data[i].pop(7)
20          parsed_data[i].pop(5)
21
22          try:
23              parsed_data[i].pop(8)
24          except Exception as p:
25              print(str(p))
```

Рисунок 3.4 Створення функції для перевірки даних з json файлу

Як наведено на рисунку 3.4, я відтворюю об’єкт parsed_data у вигляді списку. Це необхідно для легкості зчитування даних за допомогою елементів масиву. На рядку 15 відтворюю ітеративний цикл, за допомогою конструкції for in, яка буде проходити по кожній строчці лог файлу у форматі json. Для того щоб відтворити двовимірний масив на кожній ітерації додаю об’єкт пустого списку, як зазначено на рядку 16. Далі, за допомогою функції split, розбиваю об’єкт по пробілу, та видаляю зайві елементи, які не будуть використовуватися для навчання моделі. В нашому випадку це елементи статус коду у логах та кількість символів які повертаються у

HTTP відповіді сервером. Після цього створюю блок виключення, за допомогою конструкції try except і намагаюсь видалити останній елемент масиву. Блок виключення використовується для того щоб обробити ті випадки, коли логи можуть повертати неочікувані відповіді від серверу. В деяких випадках сервер може повернути повідомлення про помилку. В цьому випадку необхідно обробити дану ситуацію, щоб не виникали помилки під час навчання моделі.

```
27          try:
28              parsed_data[i][4] = int(parsed_data[i][4].split('id=')[-1])
29          except Exception as p:
30              if 'id' not in parsed_data[i][4]:
31                  parsed_data[i][4] = 1
32              else:
33                  # preparing of the dataset
34                  sql_ = ['union', 'select', 'or', 'order', 'by', 'from', 'table', 'database']
35                  is_sql = False
36                  for sql in sql_:
37                      if sql in parsed_data[i][4]:
38                          parsed_data[i][4] = 0
39                          is_sql = True
40                          break
41
42                  if not is_sql:
43                      parsed_data[i][4] = -1
44
45          parsed_data[i][3] = int(parsed_data[i][3] == 'GET')
46          parsed_data[i][5] = int(parsed_data[i][5])
47          parsed_data[i].pop(7)
48
49      return parsed_data
```

Рисунок 3.5 Підготовка dataset для навчання

Після видалення зайвих елементів, продовжуємо підготовлювати данні для нейронної мережі. Для цього будуємо ще один блок виключення з try except конструкцією та намагаємося розділити данні через вираз на рядку 28. В цьому місці уваги набуває параметр id, який при нормальній поведінці дорівнює числовому значенню. У випадку якщо даний вираз не дорівнює числовому значенню

намагаємося з’ясувати, чи не виконується якась атака. Після рядка 29 роблю перевірку на існування параметру id. У випадку якщо даного параметру в логах немає то атаки типу sql або php jnjections імовірно не відбуваються. Після цього необхідно зазначити ключові слова які можуть використовуватися для ідентифікації вразливості типу sql jnection. Сигнатурна атака sql може містити в собі одне з наступних слів: union, select, or, order, by, from, table, database. Хочу наголосити на тому що при нормальній поведінці веб серверу таких виразів при параметрі id існувати не може, тому слід зазначити, що імовірно виконується атака типу sql, що наддасть можливість своєчасний протидії або ідентифікувати інцидент безпеки.

На рядку 36 підготовлюємо данні вигляді лог – відповідь, бо нейронна мережа приймає на вхід для навчання dataset, саме у такому вигляді. Якщо говорити більш детально, в цій ітерації наша модель буде намагатися згідно сигнатурі дати вірогідність атаки, спираючись на значення з рядка 34. Для цього ми використовуємо змінну is_sql у типі boolean, при наявності ключового слова ця змінна приймає значення True та виходимо з циклу, якщо ні то вираз дорівнює -1, що означає що для даного типу виразу не буде класифікації а модель буде сприймати вираз як звичайний лог. На Строчках 45-47 виконується перевірка типу запиту HTTP. Якщо запит дорівнює GET, тоді перевірка виконується, якщо ні то пропускаємо цей лог. В такому випадку ми видаляємо лог з нашого dataset.

```
52  def main():
53      data = read_data()
54
55      parsed_data = parse_data(data)
56
57      for d in parsed_data:
58          print(d)
59
60      data_to_nn = [[parsed_data[i][3],
61                     parsed_data[i][4],
62                     parsed_data[i][5],
63                     int(parsed_data[i][7])] for i in range(len(parsed_data))]
```

Рисунок 3.6 Початок навчання моделі

На рядках 55 та 53 ми присвоюємо значення виконання функцій parse_data та read_data змінним data та parsed_data. За допомогою ітерації на рядку 57 ми виводимо на екран данні, які будуть переданні нейронній мережі, для представлення процесу. На строчці 60 підготовлюємо двовимірний масив, який зберігає екземпляр даних після перевірки. Двовимірний масив використовується лише для зручності, бо модель може приймати і одномірний, але так зручніше звертатися до об'єктів в масиві, не додаючи нумерацію всередині масиву.

```
64      # 0 - sql injection
65      # 1 - normal case
66      # 2 - cmd injection
67      res = [[1], [1], [1], [1], [0], [0], [0], [0], [2]]
68
69      # Створення екземпляру RandomForestClassifier
70      model = RandomForestClassifier(n_estimators=100, random_state=42)
71
72      # Визначення даних для навчання та цільових значень
73      X_train = data_to_nn
74      y_train = [1, 1, 1, 1, 0, 0, 0, 0, 2]
```

Рисунок 3.7 Створення екземпляра класу RandomForest

На рядках 64-67, рисунку 3.7, закоментовані значення у цифрах чому дорівнює той чи інший елемент у нашому dataset. Так як модель для навчання сприймає лише цифрові значення, маємо наступні параметрі 0 – означає атаку типу sql injection, 1 – випадок при якому не відбувається атака cmd injection та sql injection і цифра 2 означає що відбулася атака типу cmd injection. На рядку 70 створюємо екземпляр класу випадкових лісів, з використанням параметрів n_estimators, який означає кількість потоків та random_state, що означає кількість створених випадкових лісів. На рядку 73 присвоюємо значення dataset змінній x_train та y_train, що містить в собі відповіді на класифікацію наших логів у фалі data.json.

```
76      # Навчання моделі на основі датасету
77      model.fit(X_train, y_train)
78
79      # Предсказание метки класса для новых данных
80      with open('test.json', 'r') as f:
81          test_data = json.load(f)
82
83      parsed_test = parse_data(test_data)
84      test_data_to_nn = [[parsed_test[i][3],
85                          parsed_test[i][4],
86                          parsed_test[i][5],
87                          int(parsed_test[i][7])] for i in range(len(parsed_test))]
```

Рисунок 3.8 Навчання моделі

На рисунку 3.8, на рядку 77 відбувся початок тестування моделі. За допомогою функції fit, модель приймає на вхід масив з dataset та класифікацією з ідентифікацією типу атаки на застосунок на основі лог файлу. Для перевірки, що модель навчилася та працює як потрібно, ми підготували файл з логами для тестування, з назвою test.json. Тепер використовуємо функцію predict на рядку 89, і отримуємо передбачення імовірностей на рядку 92, за допомогою функції predict_proba. І вже на рядках 95 та 96 отримуємо результат виконання класифікації моделі нейронної мережі.

```
89      y_pred = model.predict(test_data_to_nn)
90
91      # Передбачення ймовірностей для кожного класу
92      y_proba = model.predict_proba(test_data_to_nn)
93
94
95      print("Передбачення: ", y_pred)
96      print("Вероігідності: ", y_proba)
```

Рисунок 3.9 Тестування моделі

Висновки за розділом 3

Отже проаналізувавши існуючи типи нейронних мереж, визначили, що для вирішення проблем класифікації даних, у нашому випадку типу атак на веб застосунки, найкраще підходить модель випадкових лісів. Використання випадкового лісу означає, що випадковий ліс добре передбачає нові дані. Це досягається завдяки бутстреповій підвибірці, що дозволяє кожному дереву бачити трохи інші дані, та випадково вибраній підмножині ознак, що забезпечує варіативність моделей. Для будування датасету, що може бути використаний нейронною мережею для навчання, необхідно підготовленні логи з веб-серверу. Для будування нашого датасету були використанні згенеровані логи, веб серверу Apache. У мові програмування python достатньо бібліотек для роботи як з датасетами так і з фреймворками нейронних моделей. Бібліотека sklearn підходить для вирішення проблем пов’язаних з класифікаціє даних, так як має вже описану модель класу RandomForest. Непотрібно переопреділяти існуючи математичні алгоритми. Лише підготувати данні для навчання, тестування та розписати нейронну мережу із заданми параметрами..

РОЗДІЛ 4
АНАЛІЗ РЕЗУЛЬТАТІВ РОБОТИ НЕЙРОННОЇ МЕРЕЖІ

Після успішної програмної реалізації моделі нейронної мережі, час проаналізувати результати. По-перше для виводу даних з прогнозування класифікацій атак на веб застосунок. був описаний клас LogTypes, використовуючи бібліотеку enum та клас Enum.

```
from logTypes import LogTypes
```

Рисунок 4.1 Описання класу LogTypes

Для описання класу на рядках 4-7 вже було обумовлено, що SQL відповідає цифрі 0, нормальна поведінка зазначається одиницею, та атака типу cmd injection зазначимо як CMD, дорівнює 2. Взагалі цей принцип з представленням атак у вигляді цифр використовується лише тому, що нейронна мережа, розуміє лише цифри, для цього ми зазначимо дану класифікація як на рисунку 4.1.

Для того щоб результати прогнозування моделі видавали зрозумілий та красивий результат ми використовуємо бібліотеку enum. Данна бібліотеку допомагає представляти у красивому термінальному вигляді.

```
1  from enum import Enum
2
3
   2 usages
4  class LogTypes(Enum):
5      SQL = 0
6      NORMAL = 1
7      CMD = 2
```

Рисунок 4.2 Імпорт LogTypes

Надалі імпортуємо описаний клас до основної частини програми. На рисунку 4.2, відбувається імпорт описаного класу. Тепер необхідно описати певну структуру коду для відображення результатів виконання програмного коду. На рисунку 4.3 наведена структура коду, з використанням функцій типу print.

```
print("Предсказания: ", y_pred)
print(f'номер тестового логу | імовірність SQL | імовірність NORMAL | імовірність CMD')

for i in range(len(y_pred)):
    print(f'{(str(i) + "-й лог").center(len("номер тестового логу "))}|'
        f'{str(y_proba[i][0]).center(len(" імовірність SQL "))}|'
        f'{str(y_proba[i][1]).center(len(" імовірність NORMAL "))}|'
        f'{str(y_proba[i][2]).center(len(" імовірність CMD "))}')
```

Рисунок 4.3 Описання конструкції виводу результатів у термінал

Після описання всіх необхідних інструкцій коду. Можна запустити програмний код та отримати результати. На рисунку 4.4 наведені результати виконання моделі нейронної мережі, яка ідентифікує з певною долею імовірнісю тип атаки на веб застосунок на основі логів, які надаються веб-сервером.

```
номер тестового логу | імовірність SQL | імовірність NORMAL | імовірність CMD
       0-й лог       |       0.02      |        0.98        |       0.0
       1-й лог       |       0.74      |        0.25        |       0.01
(env) ubuntu@Ubuntu2023Epam:~/Downloads/neuron_model_diploma$
```

Рисунок 4.4 Результати виконання коду нейронної мережі

У першому стовпці наведена нумерація логів. Для тестування використовувалися дві події у вигляді двох логів. В одному з них не виконувалась жодна атака, коли у другому була виконана атака типу SQL injection. З результатів

зрозуміло що з імовірністю 78 відсотків модель прогнозує правильну відповідь на атаку.

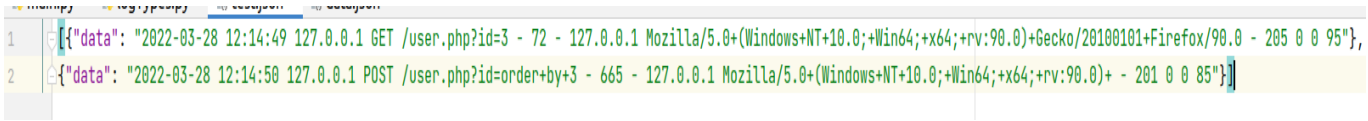

Рисунок 4.5 Контекст лог файлу, підготовленого для тестування моделі

Для тестування моделі був використаний, підготовлений лог-файл типу .json. Test.json зберігає в собі дві події веб-серверу, перший лог містить в собі нормальний запит, де id параметр дорівнює 3, звичайному числовому значенню. Другий рядок дорівнює order by 3, представлена як потенційна перевірка на sql injection. Виходячи з цього, прогнозування відбулося наступним чином: лог під номером один, модель прогнозує відсутність атаки з вірогідністю 98 відсотків. У логу під номером два прогнозується атака типу sql injection з вірогідністю 78, відсутність атаки 25 відсотків і атака типу cmd injection 1 відсоток.

Висновки за розділом 4

У 4 розділі було проведено аналіз результатів виконання роботи нейронної мережі для ідентифікації вразливостей у веб застосунках. Було доведено, що дана модель може працювати з лог-файлами, які містять інформацію про веб-сервер. На основі HTTP запитів можливо спрогнозувати імовірну атаку на застосунок. Для даної моделі був розроблений dataset, що містять реальні спроби атаки на веб-сервер. Також для тестування моделі, був зроблений тестовий лог-файл що містив в собі приклад нормальної HTTP запиту та імовірну атаку. Дана модель з високою долею імовірності спрогнозувала, що відбулася атака типу SQL injection а також з імовірністю 75 відсотків для нормального запиту, спрогнозувала, що атака не відбувалася. Було зазначено, що дана модель спроможна до навчання і може видавати імовірності більш

точно, якщо для навчання застосувати dataset більший за обсягом аніж dataset, який використовувався при реалізації моделі.

ВИСНОВКИ

У ході дипломної роботи було розв’язано завдання щодо проектування моделі нейронної мережі для ідентифікації вразливостей у веб застосунках. В ході розв’язання поставлених задач були отримані наступні наукові та практичні результати:

1. Було проведено аналіз та класифікацію загроз у веб застосунках. Досліджено основні типи атак та їх практичне застосування.

2. Досліджено та проаналізовано системи протидії атакам на веб застосунки а також досліджено існуючу модель ідентифікації вразливостей у веб застосунках Threat Modelling.

3. Було проаналізовано архітектуру веб застосунку та всі його компоненти. Досліджено існуючи типи логування подій веб серверу.

4. Реалізовано модель нейронної мережі для ідентифікації вразливостей.

5. Проаналізовано основні типи нейроних мереж.

6. Було проведено аналіз результатів роботи реалізованної моделі нейронної мережі для ідентифікації вразливостей у веб застосунках.

Після проведення аналізу архітектури веб застосунків було з’ясовано, що веб застосунки мають великий спектр атак. Також з’ясувалося, що в сучасних веб застосунках використовується модель для ідентифікації вразливостей типу Threat Modelling. Дана модель вимагає багато мануального втручання з боку інженера з кібербезпеки, що збільшує ризик неточності. Для запобігання неточностей була реалізована власна модель нейронної мережі, яка на основі подій, що відбуваються у веб застосунку, може спрогнозувати імовірність типу атаки на веб застосунок. Для реалізації моделі нейронної мережі було створено dataset на основі лог файлів, що зберігаються на боці веб серверу і містять в собі інформацію типу HTTP реквестів, тобто подій які застосовуються до веб застосунку. Модель ідентифікації вразливостей у веб застосунках вирішує проблему класифікації типів атак на додаток. У ході реалізації моделі було застосовано бібліотеку нейронного навчання Sklearn, мови

програмування Python, яка містить в собі реалізовані математичні алгоритми типу дерев рішень та рандомних лісів, які в свою чергу найліпше підходять для вирішення задач, пов’язаних з класифікацією даних. Після навчання нейронної мережі, було створено тестувальний dataset для перевірки коректності роботи моделі. Реалізована модель показала, що вона з високою долею імовірності може класифікувати атаку за типом або відсутність загрози.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Веб додатки [Електронний ресурс] UA. – 2022. – Режим доступу до ресурсу: https://blog.ithillel.ua/articles/web-application-architecture.
2. Загальні архітектури веб додатків [Електронний ресурс] Microsoft. – 2023. – Режим доступу до ресурсу: https://learn.microsoft.com/ru-ru/dotnet/architecture/modern-web-apps-azure/common-web-application-architectures.
3. Чиста архітектура веб серверів [Електронний ресурс] Хабр. – 2021. – Режим доступу до ресурсу: https://habr.com/ru/articles/493430/.
4. Чиста архітектура [Електронний ресурс] Highload. – 2019. – Режим доступу до ресурсу: https://highload.today/veb-prilozheniya/.
5. Як обрати архітектуру [Електронний ресурс] PyLoad. – 2017. – Режим доступу до ресурсу: https://blog.ithillel.ua/ru/articles/web-application-architecture.
6. Елементи архітектури веб серверу [Електронний ресурс]. – 2020. – Режим доступу до ресурсу: https://medium.com/nuances-of-programming
7. Що таке backend [Електронний ресурс] mc.today. – 2022. – Режим доступу до ресурсу: https://mc.today/chto-takoe-backend-razrabotka/.
8. Backend Development [Електронний ресурс] Highload. – 2023. – Режим доступу до ресурсу: https://highload.today/backend/.
9. Backend [Електронний ресурс], Drozd. – 2022. – Режим доступу до ресурсу: https://drozd.red/glossary/backend/.
10. Frontend or Backend? [Електронний ресурс], ItStep. – 2022. – Режим доступу до ресурсу: https://kiev.itstep.org/ru/blog/frontend-vs-backend-whats-the-difference.
11. Що таке backend розробка [Електронний ресурс], Buh24. – 2023. – Режим доступу до ресурсу: https://www.buh24.com.ua/shho-take-backend-rozrobka-i-chym-vona-vidriznyayetsya-vid-frontend/.
12. Backend для всіх [Електронний ресурс], Sky pro. – 2023. – Режим доступу до ресурсу: https://sky.pro/media/backend-razrabotchik-kto-eto-takoj-i-chem-on-zanimaetsya.

13. Backend як логіка [Електронний ресурс], Cases. – 2022. – Режим доступу до ресурсу: https://cases.media/article/khto-takii-backend-rozrobnik.
14. Frontend Developer [Електронний ресурс], ITVDN. – 2020. – Режим доступу до ресурсу: https://itvdn.com/ua/specialities/frontend-developer?admitad_uid=75a69e104cf09cb1f579d1bd33e974b3&utm_source=admitad.
15. Хто такий frontend [Електронний ресурс], ITVDN. – 2020. – Режим доступу до ресурсу: https://itvdn.com/ru/specialities/frontend-developer.
16. Фронт Енд [Електронний ресурс], Wikipedia. – 2022. – Режим доступу до ресурсу: https://ru.wikipedia.org/wiki/front_end_unicorn
17. Що таке фротенд [Електронний ресурс], Dan IT. – 2019. – Режим доступу до ресурсу: https://dan-it.com.ua/blog/razrabotka-so-storony-front-end-chto-jeto-takoe-i-chem-otlichaetsja-ot-back-end/.
18. Frontend vs Backend [Електронний ресурс], ITStep. – 2022. – Режим доступу до ресурсу: https://kiev.itstep.org/ru/blog/frontend-vs-backend-whats-the-difference.
19. Difference between Frontend and Backend [Електронний ресурс], Timeweb. – 2023. – Режим доступу до ресурсу: https://timeweb.com/ru/community/articles/frontend-i-backend-razlichiya-osobennosti-i-trebovaniya-k-specialistam.
20. Підхід до розробки [Електронний ресурс], Frontend.Kiev – Режим доступу до ресурсу: https://frontend.lviv.ua/chym-vidriznyayetsya-frontend-ta-backend-rozrobka-shho-obraty.
21. Як обрати мову програмування для веб додатку [Електронний ресурс], getit. – 2023. – Режим доступу до ресурсу: https://getit.agency/blog/difference-developers.
22. OWASP TOP TEN [Електронний ресурс], OWASP. – 2023. – Режим доступу до ресурсу: https://owasp.org/www-project-top-ten/.
23. Компанія OWASP [Електронний ресурс], PROGLIB. – 2021. – Режим доступу до ресурсу: https://proglib.io/p/chto-takoe-top-10-owasp-i-kakie-uyazvimosti-veb-prilozheniy-naibolee-opasny-2021-09-09
24. Пасічник О. М. Топ 10 актуальних вразливостей [Електронний ресурс] / Олексій Максимович Пасічник // qagroup. – 2023. – Режим доступу до ресурсу: https://qagroup.com.ua/publications/tpo-10-vulnerability-owasp/.

25. OWASP TOP-10: практический взгляд на безопасность веб-приложений [Електронний ресурс], Хабр. – 2023. – Режим доступу до ресурсу: https://habr.com/ru/companies/simplepay/articles/258499/.
26. The Open Web Application Security Project (OWASP) is a nonprofit foundation dedicated to improving software security. [Електронний ресурс], Synopsis. – 2021. – Режим доступу до ресурсу: https://www.synopsys.com/glossary/what-is-owasp-top-10.html.
27. Что такое OWASP Top-10 и как использовать указанные риски и уязвимости [Електронний ресурс], OWASP. – 2023. – Режим доступу до ресурсу: https://blog.themarfa.name/chto-takoie-owasp-top-10-i-kak-ispolzovat-ukazannyie-riski-i-uiazvimosti/.
28. OWASP Top 10 Vulnerabilities [Електронний ресурс], Veracode. – 2022. – Режим доступу до ресурсу: https://www.veracode.com/security/owasp-top-10.
29. Broken Access Control OWASP [Електронний ресурс], OWASP. – 2021. – Режим доступу до ресурсу: https://owasp.org/Top10/A01_2021-Broken_Access_Control/.
30. Все що необхідно знати про вразливість BAC [Електронний ресурс], Habr. – 2021. – Режим доступу до ресурсу: https://habr.com/ru/articles/654769/.
31. Broken Authentication [Електронний ресурс], OWASP. – 2021. – Режим доступу до ресурсу: https://owasp.org/www-project-top-ten/2017/A2_2017-Broken_Authentication.
32. A07:2021 – Identification and Authentication Failures [Електронний ресурс], OWASP. – 2021. – Режим доступу до ресурсу: https://owasp.org/Top10/A07_2021-Identification_and_Authentication_Failures/.
33. OWASP Top 10 – Broken Authentication [Електронний ресурс], code maze. – 2021. – Режим доступу до ресурсу: https://code-maze.com/owasp-broken-authentication/.
34. A02:2021 – Cryptographic Failures [Електронний ресурс], OWASP. – 2021. – Режим доступу до ресурсу: https://owasp.org/Top10/A02_2021-Cryptographic_Failures/.
35. OWASP Top 10: Cryptographic failures [Електронний ресурс], Synopsis. – 2021. – Режим доступу до ресурсу: https://www.synopsys.com/blogs/software-security/owasp-top-10-cryptographic-failures/.

36. Introduction to Cryptographic Failures [Електронний ресурс], Software Secured. – 2021. – Режим доступу до ресурсу: https://www.softwaresecured.com/introduction-to-cryptographic-failures/.

37. A02:2021 – Cryptographic Failures Owasp: Know This Cyber Trouble Better [Електронний ресурс], Wallarm. – 2021. – Режим доступу до ресурсу: https://www.wallarm.com/what/a02-2021-cryptographic-failures.

38. A03:2021 – Injection [Електронний ресурс], OWASP. – 2021. – Режим доступу до ресурсу: https://owasp.org/Top10/A03_2021-Injection/.

39. Injection Theory [Електронний ресурс]. – 2021. – Режим доступу до ресурсу: https://owasp.org/www-community/Injection_Theory.

40. SQL Injection [Електронний ресурс], OWASP. – 2021. – Режим доступу до ресурсу: https://owasp.org/www-community/attacks/SQL_Injection.

41. A1:2017-Injection [Електронний ресурс], OWASP. – 2021. – Режим доступу до ресурсу: https://owasp.org/www-project-top-ten/2017/A1_2017-Injection.

42. OWASP C. T. Injection Flaws [Електронний ресурс] / Company Trail OWASP // OWASP. – 2021. – Режим доступу до ресурсу: https://owasp.org/www-community/Injection_Flaws.

43. Injection Prevention Cheat Sheet¶ [Електронний ресурс], OWASP. – 2021. – Режим доступу до ресурсу: https://cheatsheetseries.owasp.org/cheatsheets/Injection_Prevention_Cheat_Sheet.html

44. Command Injection [Електронний ресурс], OWASP. – 2021. – Режим доступу до ресурсу: https://owasp.org/www-community/attacks/Command_Injection.

45. SQL Injection Prevention Cheat Sheet¶ [Електронний ресурс], OWASP. – 2021. – Режим доступу до ресурсу: https://cheatsheetseries.owasp.org/cheatsheets/SQL_Injection_Prevention_Cheat_Sheet.html.

46. OWASP Top 10: Injection – What It Is And How To Protect Our Applications [Електронний ресурс], OWASP. – 2021. – Режим доступу до ресурсу: https://cyolo.io/blog/owasp-top-10/owasp-top-10-injection/.

47. A04:2021 – Insecure Design [Електронний ресурс], OWASP. – 2021. – Режим доступу до ресурсу: https://owasp.org/Top10/A04_2021-Insecure_Design/.
48. Insecure Design [Електронний ресурс], CRASH TEST. – 2021. – Режим доступу до ресурсу: https://crashtest-security.com/insecure-design-vulnerability/.
49. OWASP Top 10 – #4 Insecure Design [Електронний ресурс], Foresite. – 2021. – Режим доступу до ресурсу: https://foresite.com/blog/owasp-top-10-insecure-design/.
50. OWASP Top 10: Insecure design [Електронний ресурс], Synopsis. – 2023. – Режим доступу до ресурсу: https://www.synopsys.com/blogs/software-security/owasp-top-10-insecure-design/.
51. Insecure design is a type of flaw that can sit in the background of everything you do [Електронний ресурс], Snyk. – 2023. – Режим доступу до ресурсу: https://learn.snyk.io/lessons/insecure-design/javascript/.
52. A05:2021 – Security Misconfiguration [Електронний ресурс], OWASP. – 2022. – Режим доступу до ресурсу: https://owasp.org/Top10/A05_2021-Security_Misconfiguration/.
53. A6:2017-Security Misconfiguration [Електронний ресурс], OWASP. – 2021. – Режим доступу до ресурсу: https://owasp.org/www-project-top-ten/2017/A6_2017-Security_Misconfiguration.
54. Security Misconfiguration: Impact, Examples, and Prevention [Електронний ресурс], Bright. – 2021. – Режим доступу до ресурсу: https://brightsec.com/blog/security-misconfiguration/.
55. OWASP Top 10: Security misconfiguration [Електронний ресурс], Synopsis. – 2021. – Режим доступу до ресурсу: https://www.synopsys.com/blogs/software-security/owasp-top-10-security-misconfiguration/.
56. OWASP Security Misconfiguration Vulnerability [Електронний ресурс], Indusface. – 2021. – Режим доступу до ресурсу: https://www.indusface.com/blog/owasp-security-misconfiguration/.
57. What Is a Security Misconfiguration? [Електронний ресурс], Cloud Native Wiki. – 2021. – Режим доступу до ресурсу: https://www.aquasec.com/cloud-native-academy/supply-chain-security/security-misconfigurations/.

58. OWASP Top 10 – Security Misconfigurations [Електронний ресурс], Foresite. – 2022. – Режим доступу до ресурсу: https://foresite.com/blog/owasp-top-10-security-misconfigurations/.

59. Vulnurable and Outdated Components [Електронний ресурс], OWASP. – 2022. – Режим доступу до ресурсу: https://owasp.org/Top10/A06_2021-Vulnerable_and_Outdated_Components/.

60. OWASP Top 10: #6 Vulnerable and Outdated Components [Електронний ресурс], Foresite. – 2021. – Режим доступу до ресурсу: https://foresite.com/blog/owasp-top-10-vulnerable-and-outdated-components/.

61. Vulnerable and outdated components [Електронний ресурс], Snyk. – 2023. – Режим доступу до ресурсу: https://learn.snyk.io/lessons/vulnerable-and-outdated-components/javascript/.

62. The Risks in Vulnerable and Outdated Components [Електронний ресурс], soft. – 2023. – Режим доступу до ресурсу: https://www.softwaresecured.com/the-risks-in-vulnerable-and-outdated-components/.

63. K17045144 [Електронний ресурс], MyF5. – 2022. – Режим доступу до ресурсу: https://my.f5.com/manage/s/article/K17045144.

64. A06:2021-Vulnerable and Outdated Components [Електронний ресурс], Medium. – 2022. – Режим доступу до ресурсу: https://medium.com/@shivam_bathla/a06-2021-vulnerable-and-outdated-components-a5d96017049c.

65. Comprehensive Guide to Prevent Vulnerable and Outdated Components [Електронний ресурс], OWASP. – 2020. – Режим доступу до ресурсу: https://crashtest-security.com/vulnerable-outdated-components/

66. OWASP Top 10 Deep Dive: Getting a Clear View on Vulnerable and Outdated Components [Електронний ресурс], Rapid7. – 2023. – Режим доступу до ресурсу: https://www.rapid7.com/blog/post/2021/11/08/owasp-top-10-deep-dive-getting-a-clear-view-on-vulnerable-and-outdated-components/.

67. A07:2021 – Identification and Authentication Failures [Електронний ресурс], OWASP. – 2021. – Режим доступу до ресурсу: https://owasp.org/Top10/A07_2021-Identification_and_Authentication_Failures/.

68. OWASP Top Ten – #7 Identification and Authentication Failures [Електронний ресурс], Foresite. – 2021. – Режим доступу до ресурсу: https://foresite.com/blog/owasp-top-ten-7-identification-and-authentication-failures/.
69. Identification And Authentication Failures And How To Prevent Them [Електронний ресурс], cyolo. – 2022. – Режим доступу до ресурсу: https://cyolo.io/blog/identification-and-authentication-failures-and-how-to-prevent-them/.
70. K14998322: Identification and authentication failures (A7) | Secure against the OWASP Top 10 for 2021 [Електронний ресурс], MyF5. – 2023. – Режим доступу до ресурсу: https://my.f5.com/manage/s/article/K14998322.
71. A08:2021 – Software and Data Integrity Failures [Електронний ресурс], OWASP. – 2021. – Режим доступу до ресурсу: https://owasp.org/Top10/A08_2021-Software_and_Data_Integrity_Failures/.
72. A08:2021 – Software and Data Integrity Failures- Explained [Електронний ресурс], OWASP. – 2021. – Режим доступу до ресурсу: https://crashtest-security.com/owasp-software-data-integrity-failures/.
73. A08:2021 OWASP – Software and Data Integrity Failures [Електронний ресурс], Wallarm. – 2023. – Режим доступу до ресурсу: https://www.wallarm.com/what/a04-2021-owasp-software-and-data-integrity-failures.
74. OWASP Top 10 – #8 Software and Data Integrity Failures [Електронний ресурс], Foresite. – 2021. – Режим доступу до ресурсу: https://foresite.com/blog/owasp-top-10-8-software-and-data-integrity-failures/.
75. What are software and data integrity failures? [Електронний ресурс], eductive. – 2023. – Режим доступу до ресурсу: https://www.educative.io/answers/what-are-software-and-data-integrity-failures.
76. A09:2021 – Security Logging and Monitoring Failures [Електронний ресурс], OWASP. – 2021. – Режим доступу до ресурсу: https://owasp.org/Top10/A09_2021-Security_Logging_and_Monitoring_Failures/.
77. OWASP Top Ten: #9 Security Logging and Monitoring Failures [Електронний ресурс], Foresite. – 2023. – Режим доступу до ресурсу: https://foresite.com/blog/owasp-top-ten-9-security-logging-and-monitoring-failures/.

78. What are security logging and monitoring failures? [Електронний ресурс], ed. – 2023. – Режим доступу до ресурсу: https://www.educative.io/answers/what-are-security-logging-and-monitoring-failures.

79. K94068935: Security logging and monitoring failures (A9) | Secure against the OWASP Top 10 for 2021 [Електронний ресурс], MyF5. – 2023. – Режим доступу до ресурсу: https://my.f5.com/manage/s/article/K94068935.

80. A10:2021 – Server-Side Request Forgery (SSRF) [Електронний ресурс], OWASP. – 2021. – Режим доступу до ресурсу: https://owasp.org/Top10/A10_2021-Server-Side_Request_Forgery_%28SSRF%29/.

81. OWASP Top 10 – #10 Server-Side Request Forgery [Електронний ресурс], Foresite. – 2023. – Режим доступу до ресурсу: https://foresite.com/blog/owasp-top-10-10-server-side-request-forgery/.

82. A10:2021 OWASP – Server Side Request Forgery [Електронний ресурс], Wallarm. – 2021. – Режим доступу до ресурсу: https://www.wallarm.com/what/server-side-request-forgery.

83. This Is Forgery [Електронний ресурс], code academy. – 2023. – Режим доступу до ресурсу: https://www.codecademy.com/learn/2021-owasp-top-10-server-side-request-forgery/modules/this-is-forgery/cheatsheet.

84. OWASP Top 10: The Rise of Server-Side Request Forgery (Part 1) [Електронний ресурс], Hadrian. – 2022. – Режим доступу до ресурсу: https://hadrian.io/blog/owasp-top-10-the-rise-of-server-side-request-forgery-part-1.

85. Cross Site Scripting (XSS) [Електронний ресурс], OWASP. – 2021. – Режим доступу до ресурсу: https://owasp.org/www-community/attacks/xss/.

86. Cross Site Scripting Prevention Cheat Sheet¶ [Електронний ресурс], OWASP. – 2021. – Режим доступу до ресурсу: https://cheatsheetseries.owasp.org/cheatsheets/Cross_Site_Scripting_Prevention_Cheat_Sheet.html.

87. A7:2017-Cross-Site Scripting (XSS) [Електронний ресурс], Synopsis. – 2023. – Режим доступу до ресурсу: https://owasp.org/www-project-top-ten/2017/A7_2017-Cross-Site_Scripting_(XSS).

88. Penetration Testing [Електронний ресурс]. – 2023. – Режим доступу до ресурсу: https://medium.com/svyatoslavlogyn

89. Що таке логи? [Електронний ресурс], hostiq. – 2022. – Режим доступу до ресурсу: https://hostiq.ua/wiki/ukr/log/.

90. Доступ до логів веб-сервера [Електронний ресурс], 1qb. – 2023. – Режим доступу до ресурсу: https://www.1gb.ua/services_logs.php.

91. Запуск та конфігурування [Електронний ресурс], Foresite. – 2022. – Режим доступу до ресурсу: https://docs.cipher.com.ua/pages/viewpage.action?pageId=8618139.

92. Ведение лога и ILogger [Електронний ресурс], Metanit. – 2021. – Режим доступу до ресурсу: https://metanit.com/sharp/aspnet5/2.10.php.

93. «Книга рецептів» з логування¶ [Електронний ресурс], PythonOrg. – 2023. – Режим доступу до ресурсу: https://docs.python.org/uk/3/howto/logging-cookbook.html.

94. Дзен логування. Як полюбити свої логи та почати жити [Електронний ресурс], Dou. – 2023. – Режим доступу до ресурсу: https://dou.ua/lenta/columns/about-logging/.

95. ДЕТАЛЬНІШЕ ПРО ЛОГАХ В APACHE [Електронний ресурс], APACHE. – 2022. – Режим доступу до ресурсу: http://free-review.net.ua/detalnishe-pro-logah-v-apache

96. Журнал роботи і ротація логів apache на сервері ubuntu [Електронний ресурс]. – 2023. – Режим доступу до ресурсу: https://jak.bono.odessa.ua/articles/zhurnal-roboti-i-rotacija-logiv-apache-na-serveri.php.

97. Класифікація нейронної мережі - Різні типи основних нейронних мереж [Електронний ресурс]. – 2022. – Режим доступу до ресурсу: https://uk.education-wiki.com/4761295-classification-of-neural-network.

98. Вступ. Класифікація нейронних мереж та їх властивості. [Електронний ресурс], Nure. – 2022. – Режим доступу до ресурсу: https://dl.nure.ua/pluginfile.php/634/mod_resource/content/2/content/content1.html.

99. НЕЙРОННІ МЕРЕЖІ: ЇХ ЗАСТОСУВАННЯ, РОБОТА [Електронний ресурс], Poznavayka. – 2022. – Режим доступу до ресурсу: https://www.poznavayka.org/uk/nauka-i-tehnika-2/neyronni-merezhi-yih-zastosuvannya-robota/.

100. Що таке нейронні мережі та як вони працюють? Класифікація штучних нейромереж [Електронний ресурс], Foresite. – 2020. – Режим доступу до ресурсу: https://livingfo.com/shcho-take-nejronni-merezhi-ta-iak-vony-pratsiuiut/.

101. Путівник мовою програмування Python [Електронний ресурс]. – 2020. – Режим доступу до ресурсу: https://pythonguide.rozh2sch.org.ua/.

102. МОВА ПРОГРАМУВАННЯ PYTHON — З ЧОГО РОЗПОЧАТИ НАВЧАННЯ? [Електронний ресурс], Synopsis – Режим доступу до ресурсу: https://edu.cbsystematics.com/ua/blog/python-start-blog.

103. Machine Learning in Python [Електронний ресурс], MyF5. – 2023. – Режим доступу до ресурсу: https://scikit-learn.org/stable/.

104. scikit-learn 1.2.2 [Електронний ресурс], Pip. – 2023. – Режим доступу до ресурсу: https://pypi.org/project/scikit-learn/.

105. Дерево ухвалення рішень [Електронний ресурс], Wikipedia. – 2022. – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/derevo_rishen.

106. Використання "дерева рішень" [Електронний ресурс]. – 2020. – Режим доступу до ресурсу: https://pidru4niki.com/10780621/ekonomika/vikoristannya_dereva_rishen.

107. Як працюють нейронні мережі? [Електронний ресурс], Apeps. – 2020. – Режим доступу до ресурсу: http://apeps.kpi.ua/neural-networks/en.

ДОДАТОК А

СПИСОК ОПУБЛІКОВАНИХ ПРАЦЬ ЗА ТЕМОЮ КВАЛІФІКАЦІЙНОЇ РОБОТИ

Тези наукових доповідей:

1.Олексій Шайна, Тетяна Бабенко Моделі ідентифікації вразливостей у веб застосунках. Матеріали VI Міжнародній науково-практичній конференції "Проблеми кібербезпеки інформаційно-телекомунікаційних систем (PCSITS)" (Київ, 2023)

ДОДАТОК Б

```
import json

from sklearn.ensemble import RandomForestClassifier

from logTypes import LogTypes


def read_data() -> list[dict]:
    with open("data.json", 'r') as f:
        data = json.load(f)

    return data

def parse_data(data: list[dict]) -> list[list]:
    parsed_data = []

    for i in range(len(data)):
        parsed_data.append([])

        parsed_data[i] = data[i]['data'].split(' ')
        parsed_data[i].pop(7)
        parsed_data[i].pop(5)
        try:
            parsed_data[i].pop(8)
        except:
            pass
        try:
            parsed_data[i][4] = int(parsed_data[i][4].split('id=')[-1])
        except:
            if 'id' not in parsed_data[i][4]:
                parsed_data[i][4] = 1
            else:
                # creating of the dataset
                sql_ = ['union', 'select', 'or', 'order', 'by', 'from', 'table', 'database']
                is_sql = False
                for sql in sql_:
                    if sql in parsed_data[i][4]:
                        parsed_data[i][4] = 0
                        is_sql = True
```

```
                break

            if not is_sql:
                parsed_data[i][4] = -1

        parsed_data[i][3] = int(parsed_data[i][3] == 'GET')
        parsed_data[i][5] = int(parsed_data[i][5])
        parsed_data[i].pop(7)

    return parsed_data


def main():
    data = read_data()

    parsed_data = parse_data(data)

    for d in parsed_data:
        print(d)

    data_to_nn = [[parsed_data[i][3], parsed_data[i][4], parsed_data[i][5],
int(parsed_data[i][7])] for i in range(len(parsed_data))]
    # 0 - sql
    # 1 - normal
    # 2 - php
    res = [[1], [1], [1], [1], [0], [0], [0], [0], [2]]

    # Создание экземпляра RandomForestClassifier
    model = RandomForestClassifier(n_estimators=100, random_state=42)

    # Определение данных для обучения и целевых значений
    X_train = data_to_nn # [[0, 1, 443, 200], [1, 1, 443, 200], [1, 2, 443, 200], [1, 3, 443,
200], [1, 0, 443, 200], [1, 0, 443, 200], [1, 0, 443, 200], [1, 0, 443, 200], [1, -1, 443, 200]]
    y_train = [1, 1, 1, 1, 0, 0, 0, 0, 2]

    # Обучение модели на данных
    model.fit(X_train, y_train)

    # Предсказание метки класса для новых данных
    with open('test.json', 'r') as f:
        test_data = json.load(f)

    parsed_test = parse_data(test_data)
    test_data_to_nn = [[parsed_test[i][3], parsed_test[i][4], parsed_test[i][5],
int(parsed_test[i][7])] for i in range(len(parsed_test))]
```

```
    y_pred = model.predict(test_data_to_nn)

    y_pred = [LogTypes(y_pred[i]).name for i in range(len(y_pred))]

    # Предсказание вероятностей для каждого класса
    y_proba = model.predict_proba(test_data_to_nn)



    print("Предсказания: ", y_pred)
    print(f'номер тестового логу | імовірність SQL | імовірність NORMAL | імовірність
CMD')

    for i in range(len(y_pred)):
        print(f'{(str(i) + "-й лог").center(len("номер тестового логу "))}|'
            f'{str(y_proba[i][0]).center(len(" імовірність SQL "))}|'
            f'{str(y_proba[i][1]).center(len(" імовірність NORMAL "))}|'
            f'{str(y_proba[i][2]).center(len(" імовірність CMD "))}')

if name == "main":
    main()
```