

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
НАВЧАЛЬНО-НАУКОВИЙ ФІЗИКО-ТЕХНІЧНИЙ ІНСТИТУТ
Кафедра інформаційної безпеки

До захисту допущено
Завідувач кафедри

_____ Дмитро ЛАНДЕ

(підпис)

«_____» _____ 2024 р.

Дипломна робота

на здобуття ступеня бакалавра

за освітньо-професійною програмою «Системи, технології та
математичні методи кібербезпеки»
спеціальності 125 «Кібербезпека»

на тему: Моніторинг та реагування на DDoS-атаки у реальному часі

Виконав (-ла): здобувач вищої освіти **IV** курсу, групи **ФБ-05**

(шифр групи)

_____ Даниленко Данило Андрійович

(прізвище, ім'я, по батькові)

(підпис)

Керівник: **к. т. н., доцент кафедри ІБ, Ільїн Микола Іванович**

(посада, науковий ступінь, вчене звання, прізвище, ім'я, по батькові)

(підпис)

Рецензент _____

(посада, науковий ступінь, вчене звання, науковий ступінь, прізвище, ім'я, по батькові)

(підпис)

Засвідчую, що у цій дипломній роботі немає
запозичень з праць інших авторів без
відповідних посилань.

Здобувач вищої освіти _____

(підпис)

Київ – 2024 року

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
НАВЧАЛЬНО-НАУКОВИЙ ФІЗИКО-ТЕХНІЧНИЙ ІНСТИТУТ
Кафедра інформаційної безпеки

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – 125 «Кібербезпека»

Освітньо-професійна програма «Системи, технології та математичні методи кібербезпеки»

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____ Дмитро ЛАНДЕ
(підпис)

«__» _____ 2024 р.

ЗАВДАННЯ
на дипломну роботу здобувачу вищої освіти

_____ Даниленко Данило Андрійович

(прізвище, ім'я, по батькові)

1. Тема роботи Моніторинг та реагування на DDoS-атаки у реальному часі

керівник роботи к. т. н., доцент кафедри ІБ, Ільїн Микола Іванович

(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від 31 травня 2024 р. № 2251-С

2. Термін подання здобувачем вищої освіти роботи 07.06 2024 р.

3. Вихідні дані до роботи Попередні дослідження та література на тему захисту від DDOS, зокрема рішення на основі нейронних мереж

4. Зміст роботи Огляд та властивості DDoS-атак і атаки типу Rapid Reset, огляд існуючих рішень щодо забезпечення їх безпеки, навчання та тестування моделей для виявлення атак, аналіз отриманих результатів.

5. Перелік ілюстративного матеріалу (із зазначенням плакатів, презентацій тощо)

6. Дата видачі завдання 29.01.2024

Календарний план

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів дипломної роботи	Примітка
1	Формулювання теми дипломної роботи, визначення мети та постановка задач	04.12.2023 – 12.01.2024	виконано
2	Узгодження теми	15.01.2024 – 19.01.2024	виконано
3	Визначення структури дипломної роботи	24.01.2024 – 26.01.2024	виконано
4	Отримання завдання	29.01.2024 – 30.01.2024	виконано
5	Ознайомлення з рекомендованою літературою	12.02.2024 – 12.04.2024	виконано
6	Аналіз існуючих рішень	15.04.2024 – 19.04.2024	виконано
7	Розробка підходу	22.04.2024 – 03.05.2024	виконано
8	Випробування підходу на практиці	06.05.2024 – 10.05.2024	виконано
9	Аналіз отриманих результатів	13.05.2024 – 17.05.2024	виконано
10	Оформлення дипломної роботи	01.03.2024 – 20.06.2024	виконано

Здобувач вищої освіти

_____ (підпис)

Данило Даниленко

(Власне ім'я, ПРІЗВИЩЕ)

Керівник роботи

_____ (підпис)

Микола ІЛЬІН

(Власне ім'я, ПРІЗВИЩЕ)

РЕФЕРАТ

Обсяг дипломної роботи 52 сторінки, 20 ілюстрацій, 6 таблиць і 13 джерел літератури.

Під час виконання роботи було запропоновано модель для виявлення DDoS-атак типу Rapid Reset, використовуючи сучасні методи машинного навчання та аналізу мережевого трафіку.

Для перевірки було зібрано датасет з мережевими пакетами, що включає нормальний та атакуючий трафік. Зібрані дані були оброблені та підготовлені для навчання моделей. Було застосовано різні алгоритми, такі як Random Forest і XGBoost, для виявлення атак Rapid Reset. Тестування моделей на різних наборах даних показало, що модель Random Forest дала найкращі результати з точністю 94%, тоді як модель XGBoost показала точність 56%.

Тестування на зібраному датасеті виявило, що модель Random Forest має високу ефективність у виявленні атак Rapid Reset, що підтверджується отриманими результатами. Модель Random Forest показала точність 94% з високими значеннями precision та recall для обох класів: нормального трафіку та атак.

Отримані результати: Модель здатна ефективно ідентифікувати атаки типу Rapid Reset, забезпечуючи надійний захист від DDoS-атак.

Ключові слова: DDoS, Rapid Reset, нейронні мережі, машинне навчання, кібербезпека.

ABSTRACT

The volume of the thesis is 52 pages, 20 illustrations, 6 tables and 13 sources of literature.

In the course of the work, a model was proposed to detect DDoS attacks of the Rapid Reset type using modern methods of machine learning and network traffic analysis.

For testing, a dataset with network packets was collected, including normal and attack traffic. The collected data was processed and prepared for model training. Various algorithms such as Random Forest and XGBoost were applied to detect Rapid Reset attacks. Testing the models on different datasets showed that the Random Forest model gave the best results with 94% accuracy, while the XGBoost model showed 56% accuracy.

Testing on the collected dataset showed that the Random Forest model is highly effective in detecting Rapid Reset attacks, which is confirmed by the results. The Random Forest model showed 94% accuracy with high precision and recall values for both classes: normal traffic and attacks.

Results: The model is able to effectively identify Rapid Reset attacks, providing reliable protection against DDoS attacks.

Keywords: DDoS, Rapid Reset, neural networks, machine learning, cybersecurity.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ.....	7
ВСТУП.....	9
1 ОГЛЯД ЛІТЕРАТУРИ.....	11
1.1 Класифікація DDoS-атак.....	11
1.2 Rapid Reset.....	14
ВИСНОВКИ З РОЗДІЛУ 1.....	19
2 МОДЕЛЬ ДЛЯ ЗАХИСТУ ВІД RAPID RESET АТАК.....	20
2.1 Симуляція трафіку.....	20
2.2 Збір даних.....	26
2.3 Обробка даних.....	28
2.4 Вибір моделі для нейронної мережі.....	32
ВИСНОВКИ З РОЗДІЛУ 2.....	35
3 ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ.....	35
3.1 Запуск і налаштування Docker.....	36
3.2 Навчання та тестування моделей.....	39
3.3 Результати моделей.....	43
ВИСНОВКИ З РОЗДІЛУ 3.....	48
ВИСНОВКИ.....	49
ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ.....	51

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

API (Application Programming Interface) – інтерфейс прикладного програмування, набір інструментів і протоколів для створення програмного забезпечення.

DDoS (Distributed Denial of Service) – розподілена атака відмови в обслуговуванні, мета якої – зробити сервіс або мережевий ресурс недоступним для користувачів.

HTTP/2 (Hypertext Transfer Protocol version 2) – друга версія протоколу передачі гіпертексту, що забезпечує більш ефективну передачу даних в Інтернеті.

LSTM (Long Short-Term Memory) – тип рекурентної нейронної мережі, яка здатна ефективно обробляти довгі послідовності даних і враховувати часові залежності.

ML (Machine Learning) – машинне навчання, підмножина штучного інтелекту, яка використовує статистичні методи для навчання моделей на основі даних.

Random Forest – метод ансамблевого навчання для класифікації та регресії, який використовує безліч дерев рішень для підвищення точності моделі.

Rapid Reset Attack – тип DDoS-атаки, яка використовує функції протоколу HTTP/2 для швидкого скидання потоків і створення навантаження на сервер.

SMOTE (Synthetic Minority Over-sampling Technique) – техніка синтетичного додаткового вибіркування меншості, використовується для збалансування незбалансованих датасетів.

XGBoost (Extreme Gradient Boosting) – оптимізована реалізація градієнтного бустингу, яка використовується для підвищення точності і швидкості моделі в задачах класифікації і регресії.

TLS (Transport Layer Security) – протокол безпеки транспортного рівня, що забезпечує захищену передачу даних в Інтернеті.

Wireshark – мережевий аналізатор, що дозволяє захоплювати і аналізувати мережевий трафік у реальному часі.

ВСТУП

DDoS (Distributed Denial of Service) атаки є одним із найпоширеніших та найруйнівніших видів кібератак. Вони призводять до перевантаження інфраструктури, недоступності сервісів та значних фінансових втрат. З розвитком технологій зловмисники розробляють нові методи здійснення DDoS-атак, одним з яких є Rapid Reset атака.

Rapid Reset атаки використовують швидке перезавантаження з'єднань для виснаження ресурсів сервера. Зловмисники можуть значно посилити ці атаки, оскільки невеликий обсяг трафіку здатний спричинити значний збиток. Це відбувається через те, що сервер витрачає багато ресурсів на обробку кожного запиту, що надходить під час атаки.

Мета роботи: розробка та реалізація нейронної мережі для виявлення Rapid Reset атак на основі аналізу мережевого трафіку.

Завдання роботи:

1. Аналіз існуючих методів виявлення Rapid Reset атак.
2. Розробка моделі для захисту від Rapid Reset атак.
3. Оцінка ефективності розробленої моделі.

Об'єкт дослідження: методи виявлення Rapid Reset атак.

Предмет дослідження: мережеві класифікатори для виявлення Rapid Reset атак на основі аналізу мережевого трафіку.

Методи дослідження: системний аналіз, об'єктно-орієнтоване програмування, методи машинного навчання, тестування та оцінку ефективності.

Наукова новизна одержаних результатів: запропоновано підхід до виявлення Rapid Reset атак за допомогою нейронної мережі, що базується на аналізі мережевого трафіку. Цей підхід дозволяє ідентифікувати такі атаки, використовуючи унікальні характеристики їх поведінки.

Практичне значення одержаних результатів: модель доведена до програмної реалізації, яка дозволяє виявляти трафік атак у реальному середовищі, що сприяє своєчасному виявленню та реагуванню на Rapid Reset атаки, допомагаючи оперативно реагувати на загрози.

1 ОГЛЯД ЛІТЕРАТУРИ

1.1 Класифікація DDoS-атак

DDoS-атаки (Distributed Denial of Service) є різновидом кібератак, які націлені на виведення з ладу сервісів шляхом перевантаження їх ресурсів. Існує кілька класифікацій DDoS-атак, які можна розділити за різними критеріями, такими як рівень протоколу, тип ресурсу, на який націлена атака, та техніка, що використовується для здійснення атаки. Нижче наведені основні класи та типи DDoS-атак.

1.1.1 Класифікація за рівнем протоколу

Атаки на мережевому рівні (Network Layer Attacks)

Ці атаки спрямовані на виснаження ресурсів мережевої інфраструктури, використовуючи протоколи мережевого рівня моделі OSI.

- **ICMP Flood (Ping Flood):** атака, при якій велика кількість ICMP Echo Request пакетів (Ping) надсилається на цільовий сервер, змушуючи його відповідати на кожен Echo Reply, що перевантажує мережеву смугу пропускання. [1]
- **ARP Flood:** атака, при якій зловмисник надсилає велика кількість ARP-запитів (Address Resolution Protocol) у локальну мережу, перевантажуючи мережеву інфраструктуру та обчислювальні ресурси пристроїв, що відповідають на ці запити. [2]

Атаки на транспортному рівні (Transport Layer Attacks)

Ці атаки спрямовані на виснаження ресурсів транспортного рівня, використовуючи протоколи TCP і UDP.

- **UDP Flood:** атака, при якій велика кількість UDP-пакетів надсилається на випадкові порти цільового серверу. Сервер намагається знайти додаток, пов'язаний із цими портами, але не знаходить його, що призводить до перевантаження. [3]
- **SYN Flood:** атака, при якій велика кількість SYN-запитів надсилається на цільовий сервер. Сервер відповідає SYN-ACK, але ніколи не отримує ACK від атакуючого, що призводить до виснаження ресурсів серверу. [4]

Атаки на прикладному рівні (Application Layer Attacks)

Ці атаки спрямовані на виснаження ресурсів додатків, які працюють на прикладному рівні.

- **HTTP Flood:** атака, при якій велика кількість HTTP-запитів надсилається на веб-сервер, що призводить до виснаження його ресурсів. Ця атака може бути реалізована шляхом надсилання як GET, так і POST запитів. [5]
- **Slowloris:** атака, при якій сервер тримається відкритим за допомогою неповних HTTP-запитів. Це змушує сервер тримати відкритими з'єднання, що виснажує ресурси серверу та призводить до відмови у обслуговуванні. [6]

1.1.2 Класифікація за типом ресурсу

Атаки на смугу пропускання (Bandwidth Consumption Attacks)

Ці атаки спрямовані на перевантаження смуги пропускання мережі.

- **UDP Flood:** використовує великий обсяг UDP-трафіку для перевантаження мережевих ресурсів.

- **ICMP Flood:** використовує великий обсяг ICMP-трафіку для перевантаження мережевих ресурсів.

Атаки на обчислювальні ресурси (Resource Consumption Attacks)

Ці атаки спрямовані на виснаження обчислювальних ресурсів системи.

- **SYN Flood:** використовує неповні TCP-з'єднання для виснаження ресурсів сервера.
- **HTTP Flood:** використовує великий обсяг HTTP-запитів для виснаження ресурсів веб-сервера.

Атаки на дискові ресурси (Storage Consumption Attacks)

Ці атаки спрямовані на виснаження дискових ресурсів системи.

- **Slowloris:** використовує неповні HTTP-запити для виснаження дискових ресурсів серверу.

1.1.3 Класифікація за технікою атаки

Атаки за допомогою віддзеркалення та підсилення (Reflection and Amplification Attacks)

Цей тип атак використовує відкриті сервери та послуги для збільшення обсягу трафіку, направленою на жертву.

- **DNS Amplification:** атака, при якій зловмисник надсилає запити до відкритих DNS-серверів із підробленою IP-адресою жертви. Відповіді DNS-серверів, які зазвичай є більшими за запити, надсилаються на IP-адресу жертви, що призводить до перевантаження її мережі. [7]
- **NTP Amplification:** атака, яка використовує Network Time Protocol (NTP) сервери для посилення атаки. Запити з підробленою IP-адресою жертви надсилаються до NTP серверів, що призводить до надсилання великих відповідей на адресу жертви. [8]

1.2 Rapid Reset

Rapid Reset атаки є однією з найбільш небезпечних форм DDoS-атак, яка використовує можливості протоколу HTTP/2 для виснаження ресурсів сервера шляхом швидкого скидання з'єднань. Цей розділ детально розглядає механізм дії Rapid Reset атак, їхні наслідки та методи захисту.

1.2.1 Механізм дії Rapid Reset атак

HTTP/2 був розроблений для покращення ефективності веб-трафіку, забезпечуючи мультиплексування кількох запитів і відповідей через одне TCP-з'єднання. Кожне HTTP-повідомлення розбивається на серію кадрів, що включають тип, довжину, прапори, ідентифікатор потоку (ID) і корисне навантаження. Це дозволяє одночасно обробляти кілька запитів і відповідей, що підвищує продуктивність і зменшує затримки.

Rapid Reset атаки використовують особливості протоколу HTTP/2 для створення великих навантажень на сервери. Зловмисники надсилають велику кількість фреймів RST_STREAM, що призводить до швидкого скидання HTTP/2 потоків. У протоколі HTTP/2 потік вважається закритим, як тільки відправляється кадр RST_STREAM, що дозволяє миттєво відкрити новий потік і зробити наступний запит. Це призводить до схеми запит - скидання, запит - скидання без очікування відповідей. Таким чином, зловмисники можуть надсилати неймовірну кількість запитів з однієї машини за секунду, витрачаючи значно менше ресурсів зі своєї сторони.

У серпні 2023 року Cloudflare зафіксувала атаку, під час якої було здійснено більше 201 мільйонів запитів на секунду. Ця атака була організована за допомогою ботнету з 20 000 машин. Незважаючи на порівняно невелику кількість машин, атака мала величезний вплив завдяки використанню Rapid Reset методики [9]

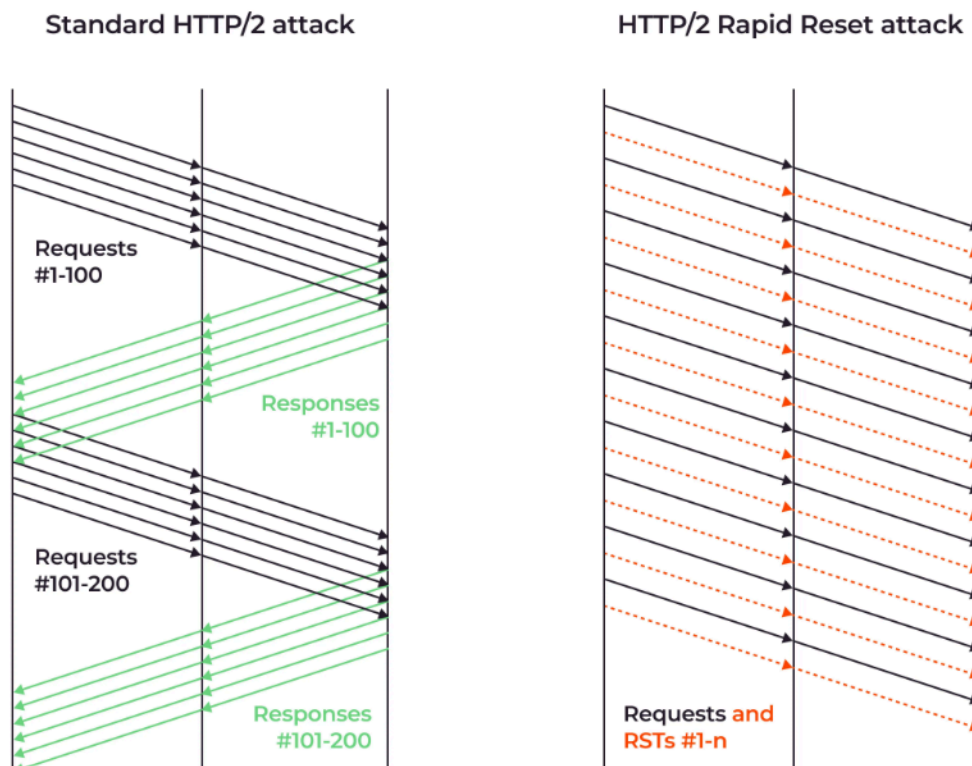


Рисунок 1.1 — Ланцюг запитів Rapid Reset та його порівняння з класичною схемою атаки [10]

1.2.2 Особливості Rapid Reset атак

Підсилення атаки

В типовій реалізації сервера HTTP/2 серверу все ще доводиться виконувати значний обсяг роботи для скасованих запитів [3], наприклад, виділяти нові структури даних потоку, аналізувати запит і виконувати розпакування заголовка, а також зіставляти URL-адресу з ресурсом. З іншого боку, клієнт практично не несе витрат за відправку запитів. Це створює корисну асиметрію витрат між сервером і клієнтом.

Вплив на системи розподілу навантаження

Сучасні системи розподілу навантаження, такі як CDN (Content Delivery Network), часто використовують асинхронні черги для обробки запитів. Rapid Reset атака порушує роботу цих черг через постійні нові запити та

скасування старих, що створює хаос у черзі. Це створює додаткове навантаження на сервери CDN і збільшує ризик їх виходу з ладу.

Додаткове навантаження на сервери бекенда

Rapid Reset атака ефективна також через те, що частина запитів, які були скасовані, все одно досягає серверів бекенда і повинні бути оброблені, хоча їх відповіді вже не потрібні. Це додатково навантажує сервери бекенда і спричиняє затримки у обробці запитів.

1.2.3 Методи захисту від Rapid Reset атак

Обмеження кількості нових потоків

Одним із методів захисту є накладення обмежень на кількість нових потоків, які можуть бути створені в одному циклі подій. Це допомагає зменшити навантаження на сервер під час атак. Наприклад, Cloudflare знизилася максимальну кількість одночасних потоків до 64, щоб зменшити вплив атак.

Контроль кадрів RST_STREAM

Контроль використання кадрів RST_STREAM і переривання з'єднання при виявленні неправомірного використання дозволяє зменшити частоту атак та їх вплив на сервер. Cloudflare запровадила моніторинг клієнтських кадрів RST_STREAM та закриття з'єднань, коли вони використовуються для зловживань.

1.2.4 Обмеження існуючих методів

Обмеження кількості нових потоків:

- Зниження максимальної кількості одночасних потоків (наприклад, до 64) може не відповідати реальним потребам легітимних користувачів, що призводить до уповільнення або збоїв у їх роботі.
- В деяких випадках може збільшувати кількість помилок типу 499.

Контроль кадрів RST_STREAM:

- Моніторинг і закриття з'єднань при виявленні зловживань можуть впливати на продуктивність і вимагати значних ресурсів для аналізу трафіку.
- Неправильна класифікація дій користувача може призвести до блокування легітимного трафіку.

1.2.5 Нейронні мережі для захисту від DDoS-атак

Вибір нейронних мереж для захисту від DDoS-атак є обґрунтованим і ефективним рішенням, що базується на їхніх унікальних можливостях аналізу великих обсягів даних, адаптивності до нових загроз та високій точності виявлення аномалій.

По-перше, нейронні мережі демонструють високу точність виявлення аномалій у трафіку, характерних для різних типів DDoS-атак. Вони здатні швидко аналізувати великі обсяги мережевого трафіку в реальному часі, що дозволяє своєчасно ідентифікувати та блокувати шкідливий трафік. Це особливо важливо для атак, які генерують величезну кількість запитів за короткий час.

Нейронні мережі здатні навчатися на нових даних, що робить їх адаптивними до змін у поведінці атакуючих. Це дозволяє їм ефективно виявляти нові методи атак і швидко адаптувати свої алгоритми для їх протидії. Завдяки цьому, захист завжди залишається актуальним і ефективним.

Крім того, нейронні мережі здатні знижувати кількість помилкових спрацьовувань, що є критично важливим для забезпечення безперервної роботи інформаційних систем. Вони використовують складні алгоритми для аналізу і навчання, що дозволяє точно визначати, які дії є легітимними, а які — ні. Це мінімізує ризик блокування легітимного трафіку і покращує користувацький досвід.

Інтелектуальна автоматизація, яку забезпечують нейронні мережі, дозволяє автоматично приймати рішення щодо блокування підозрілого трафіку. Це значно знижує навантаження на адміністраторів мережі і

підвищує швидкість реагування на атаки. Умови високого навантаження роблять швидкість і точність дій критичними для забезпечення безпеки.

Нарешті, нейронні мережі можуть аналізувати поведінкові патерни користувачів і пристроїв, що дозволяє виявляти складні і змінювані методики атак. Це робить нейронні мережі потужним інструментом для протидії складним і новим типам DDoS-атак.

Згідно з дослідженнями, нейронні мережі успішно використовуються для аналізу трафіку та захисту від DDoS-атак завдяки їх здатності до самонавчання та високої точності виявлення аномалій. Вони забезпечують надійний і ефективний захист, що дозволяє підтримувати стабільну роботу інформаційних систем і забезпечувати високий рівень кібербезпеки.

Таким чином, нейронні мережі є перспективним інструментом для захисту від DDoS-атак, завдяки своїм унікальним можливостям у виявленні та блокуванні шкідливого трафіку. [11]

ВИСНОВКИ З РОЗДІЛУ 1

У першому розділі було розглянуто кілька класифікацій DDoS-атак, що дозволяє краще розуміти різноманітні методи і підходи до проведення цих атак. Особливу увагу було приділено атакам типу Rapid Reset, які використовують особливості протоколу HTTP/2 для створення великих навантажень на сервери через швидке скидання з'єднань. Цей вид атаки дозволяє зловмисникам генерувати неймовірну кількість запитів за короткий час, витрачаючи при цьому мінімальні ресурси зі своєї сторони.

Також було детально розглянуто недоліки існуючих рішень захисту від Rapid Reset атак. Зокрема, обмеження кількості потоків, моніторинг кадрів RST_STREAM та інші методи мають свої обмеження і не завжди здатні забезпечити належний рівень захисту без негативного впливу на легітимних користувачів.

Враховуючи переваги нейронних мереж, такі як висока точність виявлення аномалій, адаптивність, швидка обробка великих обсягів даних, зниження кількості помилкових спрацьовувань та інтелектуальна автоматизація, можна зробити висновок, що використання нейронних мереж є актуальним і перспективним напрямком для захисту від Rapid Reset атак. Тому доцільним є навчання відповідної моделі для забезпечення надійного захисту інформаційних систем від подібних загроз.

2 МОДЕЛЬ ДЛЯ ЗАХИСТУ ВІД RAPID RESET АТАК

2.1 Симуляція трафіку

Для збору даних, необхідних для навчання нейронної мережі, було використано два скрипти, що генерують мережевий трафік. Один скрипт створює нормальний трафік, а інший - трафік, що імітує атаку типу Rapid Reset. Це дозволяє створити репрезентативний датасет для подальшого аналізу та навчання моделі.

2.1.1 Скрипт для симуляції нормального трафіку

Скрипт для симуляції нормального трафіку було реалізовано з використанням бібліотеки Selenium, яка автоматизує взаємодію з веб-сайтом через браузер. Цей скрипт генерує постійні запити до домашньої сторінки веб-сайту з певною періодичністю, імітуючи поведінку звичайного користувача.

```
import threading

import time

import os

from selenium import webdriver

from selenium.webdriver.chrome.options import Options

from webdriver_manager.chrome import ChromeDriverManager

from selenium.webdriver.chrome.service import Service

class WebsiteUserSelenium:

    def __init__(self):

        os.environ['SSLKEYLOGFILE'] = r"sslkeys.log"

        chrome_options = Options()

        chrome_options.add_argument("--headless")
```

```
chrome_options.add_argument("--disable-gpu")
chrome_options.add_argument("--no-sandbox")
chrome_options.add_argument("--disable-dev-shm-usage")
chrome_options.add_argument("--enable-http2")
chrome_options.add_argument("--enable-quic")
chrome_options.add_argument("--ignore-certificate-errors")
chrome_options.add_argument("--allow-insecure-localhost")
```

```
service = Service(ChromeDriverManager().install())
```

```
self.driver = webdriver.Chrome(service=service,
options=chrome_options)
```

```
def home_page(self):
    self.driver.get("https://localhost/")
    print("Home page title:", self.driver.title)
```

```
def close(self):
    self.driver.quit()
```

```
def run_test():
    user = WebsiteUserSelenium()
    try:
        while True:
            user.home_page()
```

```

        time.sleep(1)

    finally:

        user.close()

if __name__ == "__main__":
    threads = []

    for _ in range(10):

        t = threading.Thread(target=run_test)

        t.start()

        threads.append(t)

    for t in threads:

        t.join()

```

2.1.2 Скрипт для симуляції атаки Rapid Reset

Для симуляції атаки Rapid Reset використовувався скрипт, що реалізує HTTP/2 за допомогою бібліотеки h2. Цей скрипт генерує велику кількість запитів до сервера та скидає їх до отримання відповіді, створюючи навантаження на сервер.

```

import threading

import socket

import collections

import time

import random

from h2.connection import H2Connection

```

```
from h2.events import ResponseReceived, DataReceived, StreamReset
from h2.config import H2Configuration
import ssl
from h2.errors import ErrorCodes

urls = ['127.0.0.1']

methods = ['GET']

paths = ['/']

MAX_STREAMS = 5

def root_function():
    url = random.choice(urls)
    method = random.choice(methods)
    path = random.choice(paths)

    sock = socket.create_connection((url, 443))

    ctx = ssl.create_default_context()
    ctx.check_hostname = False
    ctx.verify_mode = ssl.CERT_NONE
```

```
ctx.set_alpn_protocols(['h2'])

sock = ctx.wrap_socket(sock, server_hostname=url)

assert sock.selected_alpn_protocol() == 'h2'

config = H2Configuration(client_side=True)
conn = H2Connection(config=config)
conn.initiate_connection()
sock.sendall(conn.data_to_send())

active_streams = set()

while True:
    if len(active_streams) < MAX_STREAMS:
        stream_id = conn.get_next_available_stream_id()
        headers = [
            (':method', method),
            (':authority', url),
            (':path', path),
            (':scheme', 'https')
        ]
        conn.send_headers(stream_id, headers)
        sock.sendall(conn.data_to_send())
```

```

    active_streams.add(stream_id)

# Read some data
data = sock.recv(65535)

if not data:
    break

events = conn.receive_data(data)

for event in events:
    if isinstance(event, ResponseReceived):
        print(f"Stream {event.stream_id} received response.")
    elif isinstance(event, DataReceived):
        print(f"Stream {event.stream_id} received data.")
        conn.reset_stream(event.stream_id,
error_code=ErrorCodes.CANCEL)
    elif isinstance(event, StreamReset):
        print(f"Stream {event.stream_id} reset.")
        active_streams.discard(event.stream_id)

sock.sendall(conn.data_to_send())

time.sleep(random.uniform(0.1, 2.0))

threads = []

for i in range(50):

```

```

thread = threading.Thread(target=root_function)

thread.start()

threads.append(thread)

```

```

for thread in threads:

```

```

    thread.join()

```

Скрипт налаштовує TCP-з'єднання, яке обгортається в TLS для забезпечення безпеки, і використовує HTTP/2. Кожен потік генерує запити до сервера та скидає їх до отримання відповіді.

2.2 Збір даних

Для коректного навчання моделі нейронної мережі необхідно було зібрати великий обсяг даних мережевого трафіку. Цей процес включав захоплення трафіку під час роботи двох скриптів: одного для генерації нормального трафіку, іншого для симуляції атаки Rapid Reset. Для збору даних використовувався інструмент Wireshark, який дозволяє аналізувати та зберігати мережеві пакети.

2.2.1 Використання Wireshark для збору даних

Wireshark є потужним інструментом для захоплення та аналізу мережевого трафіку. Він дозволяє фільтрувати, декодувати та аналізувати пакети, що проходять через мережу. Для збору даних під час симуляції трафіку Wireshark використовувався для запису всіх пакетів, що надходять на сервер.

2.2.2 Захоплення зашифрованого трафіку

Оскільки весь трафік між клієнтом і сервером був зашифрований за допомогою TLS, необхідно було забезпечити можливість розшифрування пакетів для їх подальшого аналізу. Для цього всі скрипти, що генерували трафік, зберігали ключі шифрування в файл `sslkeys.log`. Цей файл надавався Wireshark для розшифрування зашифрованого трафіку.

```
os.environ['SSLKEYLOGFILE'] = r"sslkeys.log"
```

Рисунок 2.2 — Фрагмент коду для визначення файлу ключів шифрування

2.2.3 Процес збору даних

1. Запуск скриптів:

- Обидва скрипти (для нормального трафіку і для атаки) запускалися по черзі, створюючи різні сценарії трафіку.
- Кожен скрипт зберігав ключі шифрування в файл `sslkeys.log`.

2. Налаштування Wireshark:

- Wireshark налаштовувався на прослуховування відповідного мережевого інтерфейсу для захоплення всього вхідного і вихідного трафіку.
- В Wireshark було завантажено файл `sslkeys.log` для розшифрування зашифрованих TLS пакетів.
- Використовувалися фільтри для захоплення тільки релевантного трафіку (HTTP/2 запити та відповіді).

3. Аналіз і збереження даних:

- Захоплені пакети зберігалися у форматі PCAP для подальшого аналізу.
- Використовувалися додаткові фільтри та інструменти Wireshark для виділення необхідних даних та аналізу потрібних аспектів трафіку.

2.2.4 Перетворення даних у формат CSV

Для подальшого аналізу і використання в навчанні моделі нейронної мережі дані з формату PCAP були перетворені у формат CSV. Це дозволило спростити обробку та аналіз даних.

```
tshark -r attack_traffic.pcapng -Y http2 -T fields -e frame.number -e
frame.time -e frame.len -e ip.src -e ip.dst -e _ws.col.Protocol -e tcp.srcport -e
tcp.dstport -e http2.streamid -e http2.type -e http2.length -e http2.flags -e
_ws.col.Info -E header=y -E separator=, -E quote=d > http2_traffic.csv
```

Для перетворення файлу захопленого трафіку attack_traffic.pcap у формат CSV використовувалася команда tshark. Вона вказує на вхідний файл формату PCAP, що містить захоплений трафік (-r attack_traffic.pcap), і фільтрує трафік, відбираючи тільки пакети HTTP/2 (-Y http2). Вивід повинен бути у форматі полів (-T fields). Команда також вказує на поля, які необхідно витягнути з кожного пакету: номер кадру, час, довжина, IP-адреси, протокол, порти, ID потоку, тип, довжина, прапори та інформація (-e). Додаткові параметри (-E header=y -E separator=, -E quote=d) забезпечують додавання заголовків колонок, використання коми як роздільника полів і двійні лапки для текстових полів.

```
frame.number,frame.time,frame.len,ip.src,ip.dst,_ws.col.protocol,tcp.srcport,tcp.dstport,http2.streamid,http2.type,http2.length,http2.flags,_ws.col.info
"21651","Jun 12, 2024 20:08:58.178840000 Фінляндія (лето)","113","127.0.0.1","127.0.0.1","HTTP2","443","43054","0,0","4,8","18,4","0x00,0x00","SETTINGS[0], WINDOW_UPDATE[0]"
"21667","Jun 12, 2024 20:08:58.182133000 Фінляндія (лето)","355","127.0.0.1","127.0.0.1","HTTP2","443","43049","0,0","4,8","18,4","0x00,0x00","SETTINGS[0], WINDOW_UPDATE[0]"
"21689","Jun 12, 2024 20:08:58.188400000 Фінляндія (лето)","113","127.0.0.1","127.0.0.1","HTTP2","443","43046","0,0","4,8","18,4","0x00,0x00","SETTINGS[0], WINDOW_UPDATE[0]"
"21699","Jun 12, 2024 20:08:58.191748000 Фінляндія (лето)","355","127.0.0.1","127.0.0.1","HTTP2","443","43044","0,0","4,8","18,4","0x00,0x00","SETTINGS[0], WINDOW_UPDATE[0]"
```

Рисунок 2.3 — Приклад пакетів в csv файлі

2.3 Обробка даних

Для ефективного навчання нейронної мережі на основі зібраного трафіку було необхідно виконати ретельну обробку даних. Цей процес включав кілька ключових етапів: очищення даних, трансформацію, кодування, розбиття на частини, перевірку балансу класів і застосування методів балансування.

2.3.1 Очищення та обробка часових міток

Часові мітки у даних потребували очищення та перетворення для коректного аналізу. Спочатку часові мітки очищаються від зайвих символів, а потім перетворювалися в UNIX-час.

```
def clean_timestamp(timestamp):
    return re.sub(r' Финляндия \(\лето\)', '', timestamp)
```

Рисунок 2.4 — Функція для очищення часових міток від тексту

```
def convert_to_unix_time(timestamp):
    try:
        timestamp = re.sub(r' \.\d{3}\d*$', '', timestamp)
        return datetime.strptime(timestamp, '%b %d, %Y %H:%M:%S.%f').timestamp()
    except ValueError:
        try:
            return datetime.strptime(timestamp, '%b %d, %Y %H:%M:%S').timestamp()
        except Exception as e:
            return None
```

Рисунок 2.5 — Функція для перетворення мітки в формат UNIX

Для забезпечення цілісності даних відсутні значення замінювалися середніми значеннями відповідних стовпців.

```
def fix_none_timestamps(data, mean_time):
    data['frame.time'] = data['frame.time'].apply(lambda x: mean_time if pd.isna(x) else x)
    return data

mean_time_attack = attack_traffic['frame.time'].mean()
mean_time_normal = normal_traffic['frame.time'].mean()

attack_traffic = fix_none_timestamps(attack_traffic, mean_time_attack)
normal_traffic = fix_none_timestamps(normal_traffic, mean_time_normal)
```

Рисунок 2.6 — Заміна відсутніх значень середніми

2.3.2 Кодування категоріальних змінних

Текстові дані були перетворені в числовий формат за допомогою LabelEncoder, що дозволило моделі працювати з текстовими даними.

Відсутні числові значення замінювалися середніми значеннями відповідних стовпців.

```
label_encoder_attack = LabelEncoder()
label_encoder_normal = LabelEncoder()

string_cols = attack_traffic.select_dtypes(include=['object']).columns

for col in string_cols:
    if not attack_traffic[col].empty:
        most_frequent_value_attack = attack_traffic[col].value_counts().idxmax()
    else:
        most_frequent_value_attack = 'unknown'
    if not normal_traffic[col].empty:
        most_frequent_value_normal = normal_traffic[col].value_counts().idxmax()
    else:
        most_frequent_value_normal = 'unknown'
    attack_traffic[col].fillna(most_frequent_value_attack, inplace=True)
    normal_traffic[col].fillna(most_frequent_value_normal, inplace=True)

for col in string_cols:
    attack_traffic[col] = label_encoder_attack.fit_transform(attack_traffic[col])
    normal_traffic[col] = label_encoder_normal.fit_transform(normal_traffic[col])
```

Рисунок 2.5 — Кодування категоріальних змінних

```
numeric_cols = attack_traffic.select_dtypes(include=['float64', 'int64']).columns
attack_traffic[numeric_cols] = attack_traffic[numeric_cols].apply(pd.to_numeric, errors='coerce')
normal_traffic[numeric_cols] = normal_traffic[numeric_cols].apply(pd.to_numeric, errors='coerce')

attack_traffic.fillna(attack_traffic.mean(), inplace=True)
normal_traffic.fillna(normal_traffic.mean(), inplace=True)
```

Рисунок 2.6 — Заміна середніми значеннями

2.3.3 Розбиття даних на частини

Для комплексного аналізу трафіку дані були розбиті на частини. Це дозволило враховувати контекст та послідовність запитів, що є важливим при

детектуванні атак. Одного пакета може бути недостатньо для виявлення атаки, тому фрагментація суттєво покращує якість аналізу.

```
def chunk_data(data, chunk_size=10):
    chunks = [data.iloc[i:i + chunk_size] for i in range(0, len(data), chunk_size)]
    return chunks

attack_chunks = chunk_data(attack_traffic)
normal_chunks = chunk_data(normal_traffic)

labeled_attack_chunks = [(chunk.mean(axis=0), 1) for chunk in attack_chunks if len(chunk) == 10] # 1 для attack
labeled_normal_chunks = [(chunk.mean(axis=0), 0) for chunk in normal_chunks if len(chunk) == 10] # 0 для normal

all_labeled_chunks = labeled_attack_chunks + labeled_normal_chunks
random.shuffle(all_labeled_chunks)
```

Рисунок 2.7 — Розбиття даних на частини

2.3.3 Перевірка балансу класів та SMOTE

Для забезпечення рівномірного розподілу між нормальним трафіком та трафіком атак була перевірена рівномірність розподілу міток.

```
Розподіл міток даних:
1      508
0      343
Name: label, dtype: int64
```

Рисунок 2.8 — Розподіл міток даних

Результат перевірки показав, що дані не були достатньо збалансованими

Для вирівнювання кількості прикладів різних класів був використаний метод SMOTE (Synthetic Minority Over-sampling Technique). SMOTE дозволяє створювати синтетичні приклади для менш представленого класу, тим самим збалансовуючи датасет.

SMOTE працює шляхом створення нових синтетичних зразків на основі існуючих даних меншості. Це досягається шляхом обчислення векторів між існуючими зразками і генерації нових точок уздовж цих векторів.

Таким чином, після застосування SMOTE дані стали збалансованими, що покращує якість навчання моделі нейронної мережі.

2.4 Вибір моделі для нейронної мережі

При виборі моделі для задачі класифікації атак типу Rapid Reset слід враховувати кілька факторів, включаючи характеристики даних, вимоги до інтерпретованості, швидкість навчання і передбачення, а також можливість роботи з незбалансованими даними. Нижче представлена загальна таблиця з основними моделями, які можуть бути використані для цієї задачі, з вказівкою їхніх переваг та недоліків.

Таблиця 2.1 Порівняння моделей

Модель	Переваги	Недоліки
Random Forest	Добре працює з різномірними даними Стійкий до викидів та шуму; Здатна обробляти пропущені значення Вбудована оцінка важливості ознак	Може бути повільним при великій кількості дерев Менш інтерпретований у порівнянні з окремими деревами Може перевчитися на малих вибірках
Gradient Boosting (XGBoost, LightGBM, CatBoost)	Висока точність на багатьох задачах Здатна обробляти незбалансовані дані Може обробляти пропущені значення Добре масштабується на великі дані	Тривалий час навчання на великих даних Вимагає ретельного налаштування гіперпараметрів Висока складність моделі
Logistic Regression	Простота реалізації та інтерпретації Швидке навчання та передбачення Добре працює на лінійно роздільних даних	Обмежена потужність для нелінійних залежностей Менша точність на складних даних Чутлива до викидів та шуму

Кінець Таблиці 2.1

Neural Networks (LSTM, CNN)	Висока потужність для складних залежностей Здатність виявляти складні патерни в даних Гнучкість в архітектурі моделі Може обробляти тимчасові ряди (LSTM)	Тривалий час навчання і передбачення Вимагає великого обсягу даних для навчання Висока обчислювальна складність Складність налаштування і інтерпретації
Support Vector Machine (SVM)	Висока точність на малих і середніх наборах даних Добре працює з високою розмірністю ознак Здатна обробляти нелінійні залежності через ядра	Неефективний на великих наборах даних Вимагає масштабування ознак Висока обчислювальна складність при великих обсягах даних

Для задачі класифікації атак типу Rapid Reset, враховуючи особливості даних (наявність часових залежностей, високий дисбаланс класів), найбільш підходящими моделями є **Random Forest** і **Gradient Boosting** (наприклад, XGBoost). Ці моделі добре справляються з незбалансованими даними, мають високу точність і надають можливості для оцінки важливості ознак, що допомагає в інтерпретації результатів. **Neural Networks** (наприклад, LSTM) також можуть бути використані, особливо якщо необхідно враховувати часові залежності в даних, проте вони вимагають великих обсягів даних і ресурсів для навчання.

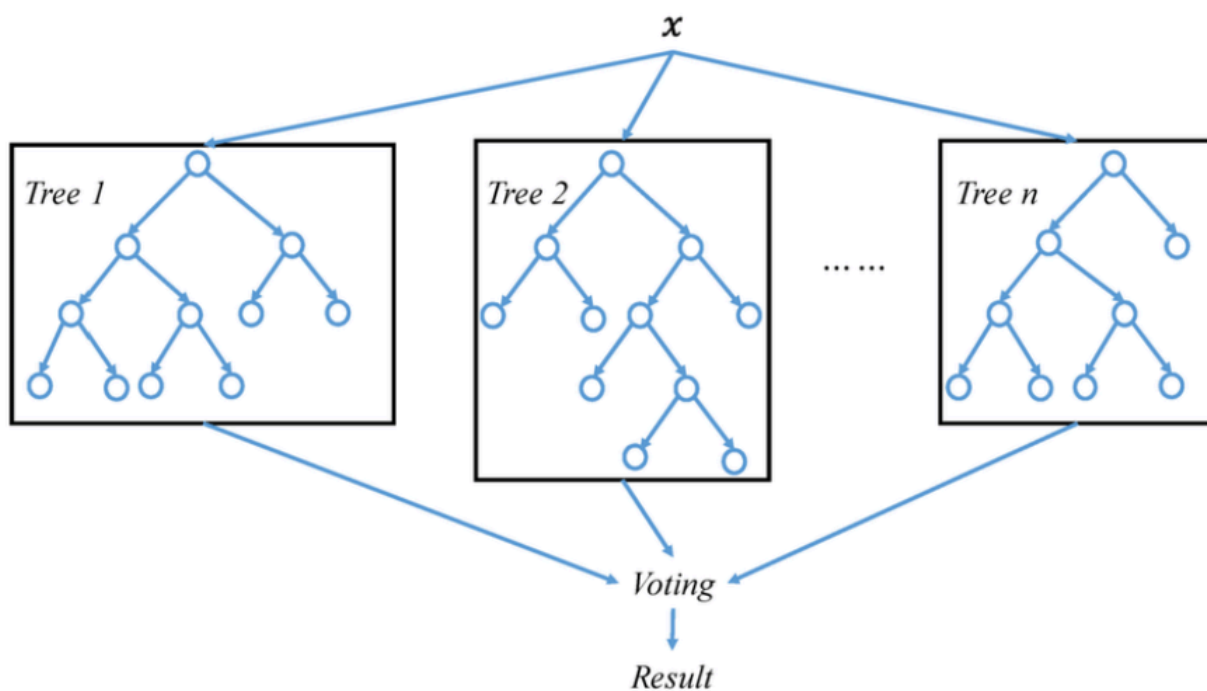


Рисунок 2.8 — A general architecture of Random Forest [12]

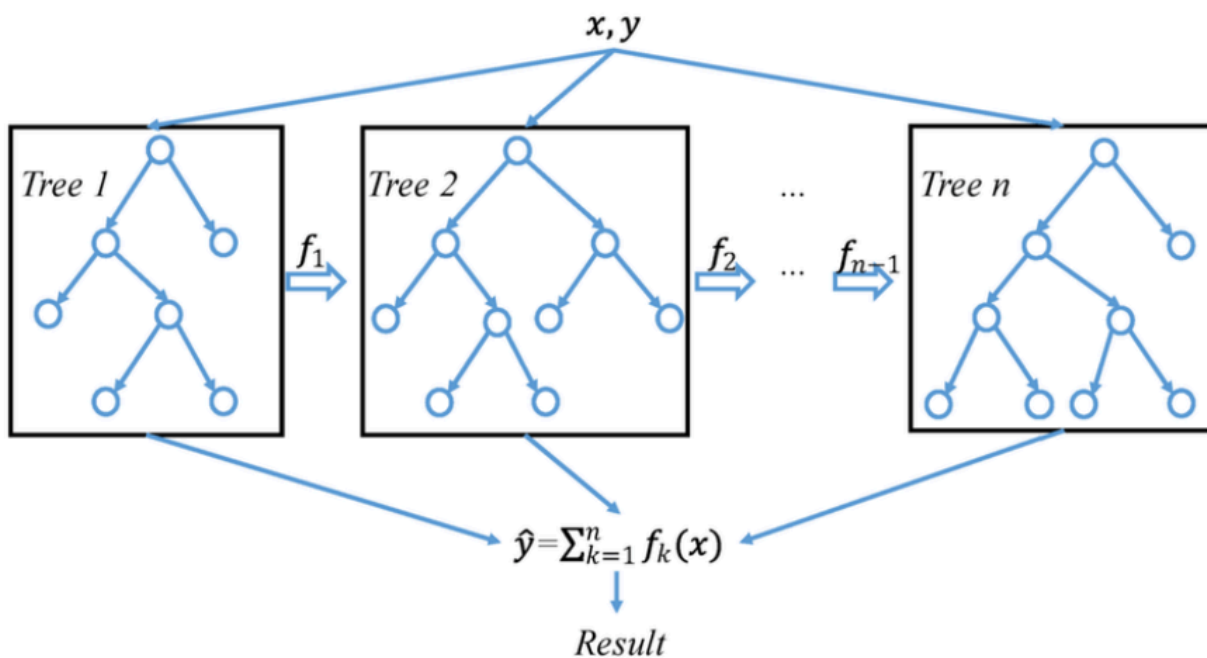


Рисунок 2.9 — A general architecture of XGBoost [13]

ВИСНОВКИ З РОЗДІЛУ 2

У другому розділі було детально розглянуто всі етапи підготовки та обробки даних для навчання нейронної мережі з метою виявлення атак типу Rapid Reset. Процес симуляції трафіку забезпечував створення як нормального, так і атакуючого трафіку, який був зібраний за допомогою Wireshark. Завдяки використанню SSL ключів для розшифровки TLS трафіку вдалося отримати точні дані про запити та відповіді серверів. Зібрані дані були ретельно очищені та перетворені для аналізу, включаючи обробку тимчасових міток, заповнення відсутніх значень і кодування категоріальних змінних.

Особливу увагу було приділено розбиттю даних на частини для врахування контексту та послідовності запитів, що є важливим для детекції атак. Це дозволило враховувати взаємозв'язки між пакетами, що значно покращило якість аналізу. Балансування даних за допомогою методу SMOTE дозволило вирівняти кількість прикладів різних класів, забезпечуючи рівномірний розподіл даних для навчання.

При виборі моделі для навчання були проаналізовані різні підходи, і було вирішено використовувати моделі Random Forest і Gradient Boosting (XGBoost), які добре справляються з незбалансованими даними і забезпечують високу точність. Крім того, для врахування тимчасових залежностей була розглянута можливість використання LSTM, яка також може бути ефективною для виявлення атак типу Rapid Reset.

Таким чином, вибраний підхід до підготовки та обробки даних, а також ретельний вибір моделі нейронної мережі забезпечують ефективне навчання системи для виявлення атак Rapid Reset у мережевому трафіку. Це створює надійну основу для подальшого розгортання та використання системи у реальних умовах для забезпечення безпеки мережевої інфраструктури.

3 ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ

3.1 Запуск і налаштування Docker

Для коректного навчання моделі та збору відповідного датасету необхідно мати сервер для взаємодії. У зв'язку з цим було прийнято рішення підняти Docker-контейнер з простим веб-сервером на Flask, що дозволяє створити ізольоване середовище для симуляції трафіку і збору даних.

3.1.1 Вибір базового образу та встановлення необхідних пакетів

```
FROM ubuntu:18.04

RUN apt-get update && apt-get install -y \
    build-essential \
    libpcre3 \
    libpcre3-dev \
    zlib1g \
    zlib1g-dev \
    libssl-dev \
    openssl \
    vim \
    wget \
    tar \
    python3 \
    python3-pip \
```

Рисунок 3.1 — Фрагмент коду з вказанням версії системи

Для створення стабільного середовища було обрано базовий образ `ubuntu:18.04`. Цей дистрибутив є широко використовуваним та підтримуваним, що забезпечує сумісність з більшістю інструментів та бібліотек. Встановлення необхідних пакетів виконується за допомогою команди `apt-get`, яка оновлює список пакетів і встановлює такі важливі залежності, як `build-essential`, `libpcre3`, `zlib1g`, `libssl-dev`, `openssl`, `python3`, `python3-pip` та інші. Ці пакети необхідні для компіляції та налаштування Nginx з підтримкою HTTP/2 та SSL, а також для роботи з мережею та тестуванням.

3.1.2 Установка Flask і Gunicorn

Flask було обрано як мікрофреймворк для розробки веб-додатків через його легкість та гнучкість. Gunicorn використовується як WSGI HTTP-сервер для забезпечення надійної роботи Flask-додатків у виробничому середовищі. Пакети встановлюються за допомогою `pip3`.

3.1.3 Скачування та компіляція Nginx з підтримкою HTTP/2

```
RUN wget https://nginx.org/download/nginx-1.21.3.tar.gz -O /tmp/nginx.tar.gz \
    && tar -xvzf /tmp/nginx.tar.gz -C /tmp

WORKDIR /tmp/nginx-1.21.3
RUN ./configure --with-http_ssl_module --with-http_v2_module \
    && make \
    && make install
```

Рисунок 3.2 — Фрагмент коду з завантаженням nginx

Скачування вихідного коду Nginx здійснюється з офіційного сайту, після чого виконується його розпакування та компіляція з підтримкою HTTP/2

(--with-http_v2_module) та SSL (--with-http_ssl_module). Це забезпечує можливість обслуговування безпечних з'єднань і підвищує ефективність роботи сервера.

3.1.4 Файл конфігурації Nginx

У цій частині буде розглянуто налаштування конфігураційного файлу Nginx, що використовується для забезпечення коректної роботи веб-сервера в контейнері Docker. Конфігурація включає налаштування для обробки HTTP та HTTPS запитів з підтримкою HTTP/2.

```
worker_processes 1;

events {
    worker_connections 1024;
}

http {
    upstream backend {
        server 0.0.0.0:5000;
    }

    server {
        listen 8081;
        server_name localhost;

        location / {
            proxy_pass http://backend;
            proxy_set_header Host $host;
            proxy_set_header X-Real-IP $remote_addr;
            proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
            proxy_set_header X-Forwarded-Proto $scheme;
        }
    }
}
```

```

server {
    listen 8443 ssl http2;
    server_name localhost;

    ssl_certificate /etc/ssl/certs/nginx-selfsigned.crt;
    ssl_certificate_key /etc/ssl/private/nginx-selfsigned.key;

    location / {
        proxy_pass http://backend;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto $scheme;
    }
}

```

Файл конфігурації Nginx включає налаштування для двох серверних блоків: один для обслуговування HTTP-запитів на порту 8081 і другий для HTTPS-запитів на порту 8443 з використанням HTTP/2. Використання HTTP/2 дозволяє відтворити специфіку Rapid Reset атак, що є необхідним для збору відповідного датасету.

У першому серверному блоці, що відповідає за обробку HTTP-запитів, Nginx налаштований на прослуховування порту 8081. Запити, що надходять на цей порт, проксуються до бекенд-сервера, визначеного в блоці `upstream`.

Другий серверний блок налаштований на обробку HTTPS-запитів на порту 8443 з використанням HTTP/2. Це забезпечує безпечне і швидке з'єднання завдяки шифруванню SSL та можливостям HTTP/2. Для цього було створено

самопідписаний SSL сертифікат, який використовується для шифрування трафіку між клієнтом і сервером

3.2 Навчання та тестування моделей

Після підготовки і обробки даних наступним етапом стало навчання і тестування обраних моделей для виявлення атак типу Rapid Reset. У цьому розділі буде детально описано процес навчання моделей Random Forest і Gradient Boosting (XGBoost), а також результати їх тестування.

3.2.1 Навчання моделі Random Forest

Модель Random Forest була обрана завдяки її здатності ефективно працювати з незбалансованими даними і високій стійкості до перенавчання. Для навчання цієї моделі було використано 100 дерев рішень, що забезпечило високу точність і надійність класифікації.

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report, accuracy_score
import time

rf_model = RandomForestClassifier(n_estimators=100, random_state=42)
start_time = time.time()
rf_model.fit(X_train, y_train)
training_time_rf = time.time() - start_time

start_time = time.time()
y_pred_rf = rf_model.predict(X_test)
prediction_time_rf = time.time() - start_time

print(classification_report(y_test, y_pred_rf))
print("Accuracy:", accuracy_score(y_test, y_pred_rf))
print(f"Training time: {training_time_rf:.4f} seconds")
print(f"Prediction time: {prediction_time_rf:.4f} seconds")
```

Рисунок 3.3 — Навчання моделі Random Forest

Таблиця 3.1 Навчання Random Forest

	precision	recall	f1-score	support
0	0.97	0.97	0.97	109
1	0.97	0.97	0.97	95
accuracy			0.97	204
macro avg	0.97	0.97	0.97	204
weighted avg	0.97	0.97	0.97	204

Accuracy: 0.970

Training time: 0.1223 seconds

Prediction time: 0.0065 seconds

3.2.2 Навчання моделі Gradient Boosting (XGBoost)

Модель XGBoost була обрана завдяки її високій продуктивності і здатності працювати з великими обсягами даних. Ця модель використовує метод градієнтного бустингу для покращення якості класифікації, поступово додаючи нові дерева рішень до ансамблю.

```

import xgboost as xgb
import time

xgb_model = xgb.XGBClassifier(random_state=42)
start_time = time.time()
xgb_model.fit(X_train, y_train)
training_time_xgb = time.time() - start_time

start_time = time.time()
y_pred_xgb = xgb_model.predict(X_test)
prediction_time_xgb = time.time() - start_time

print(classification_report(y_test, y_pred_xgb))
print("Accuracy:", accuracy_score(y_test, y_pred_xgb))
print(f"Training time: {training_time_xgb:.4f} seconds")
print(f"Prediction time: {prediction_time_xgb:.4f} seconds")

```

Рисунок 3.4 — Навчання моделі XGBoost

Таблиця 3.2 Навчання XGBoost

	precision	recall	f1-score	support
0	0.97	0.96	0.97	109
1	0.97	0.97	0.97	95
accuracy			0.97	204
macro avg	0.97	0.97	0.97	204
weighted avg	0.97	0.97	0.97	204

Accuracy: 0.965

Training time: 0.0391 seconds

Prediction time: 0.0010 seconds

3.3 Результати моделей

Після навчання та тестування моделей Random Forest і Gradient Boosting (XGBoost) були отримані наступні результати на нових даних, що підтверджують їх ефективність та можливості виявлення атак типу Rapid Reset.

3.3.1 Результати моделі Random Forest

Модель Random Forest продемонструвала високу точність класифікації на нових даних. Точність моделі становила 0.94, що свідчить про її здатність ефективно виявляти атаки Rapid Reset.

Таблиця 3.3 Результат Random Forest на нових даних

	precision	recall	f1-score	support
0	0.89	0.97	0.94	93
1	0.97	0.89	0.93	99
accuracy			0.94	192
macro avg	0.95	0.94	0.94	192
weighted avg	0.95	0.94	0.94	192

Accuracy: 0.9427

Prediction time: 0.0075 seconds

Результати показали, що модель має високу точність (precision) і повноту (recall) для обох класів: нормального трафіку та атак. Час передбачення склав всього 0.0075 секунд, що дозволяє використовувати її в реальному часі.

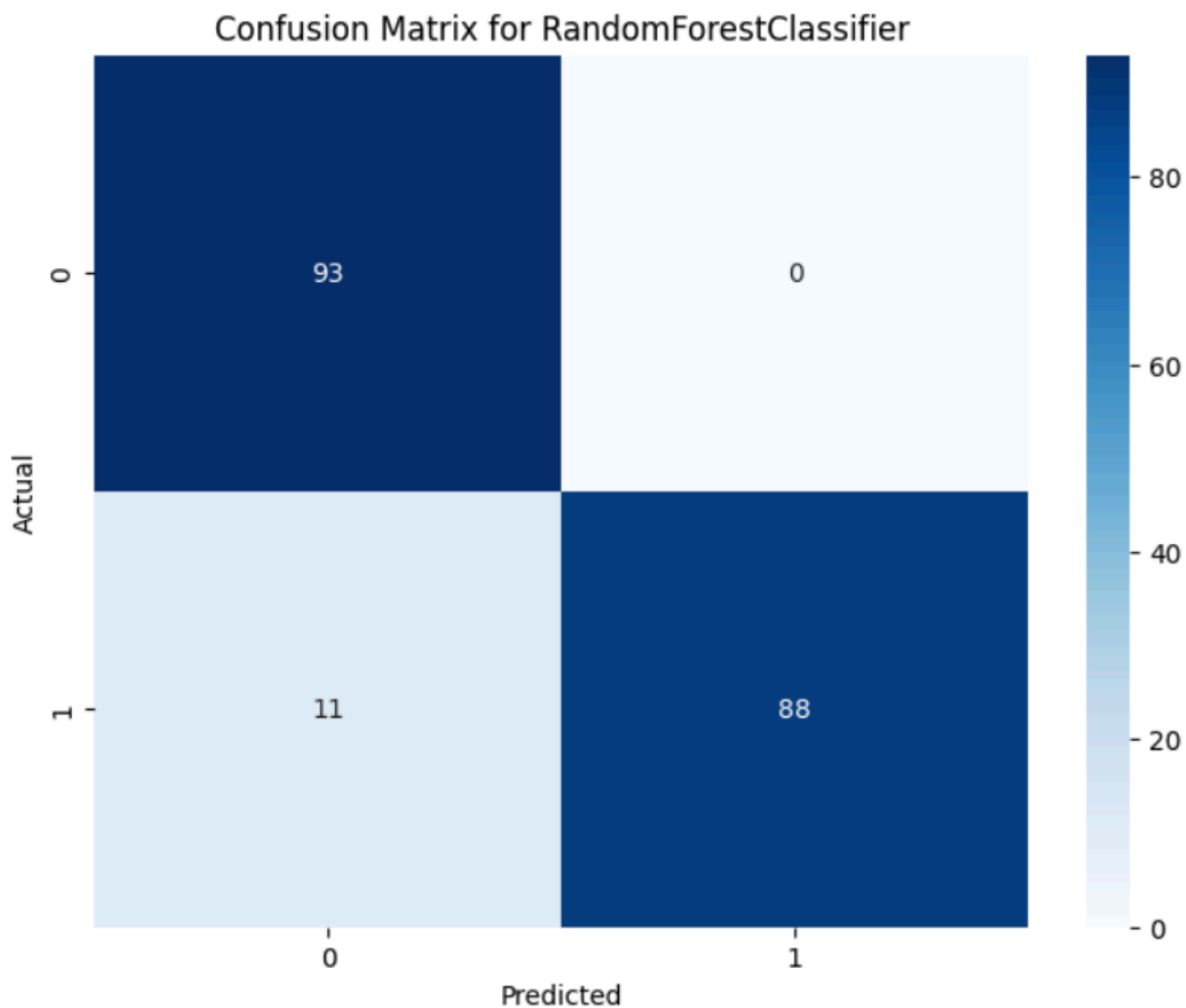


Рисунок 3.5 — Матриця плутанини моделі Random Forest

3.3.2 Результати моделі Gradient Boosting (XGBoost)

Модель XGBoost продемонструвала меншу точність у порівнянні з Random Forest, з точністю 0.56. Це свідчить про те, що модель може мати труднощі з виявленням атак Rapid Reset на нових даних.

Таблиця 3.4 Результат XGBoost на нових даних

	precision	recall	f1-score	support
0	0.53	0.97	0.69	93
1	0.97	0.15	0.26	99
accuracy			0.56	192

Кінець Таблиці 3.4

macro avg	0.76	0.58	0.48	192
weighted avg	0.77	0.56	0.47	192

Accuracy: 0.5625

Prediction time: 0.0010 seconds

Хоча модель XGBoost має високу точність для класу атак (0.97), вона показала низьку повноту (0.15), що призводить до низького значення F1-міри (0.26) для атак. Це може вказувати на те, що модель має високий рівень хибних негативних результатів, тобто не виявляє всі атаки.

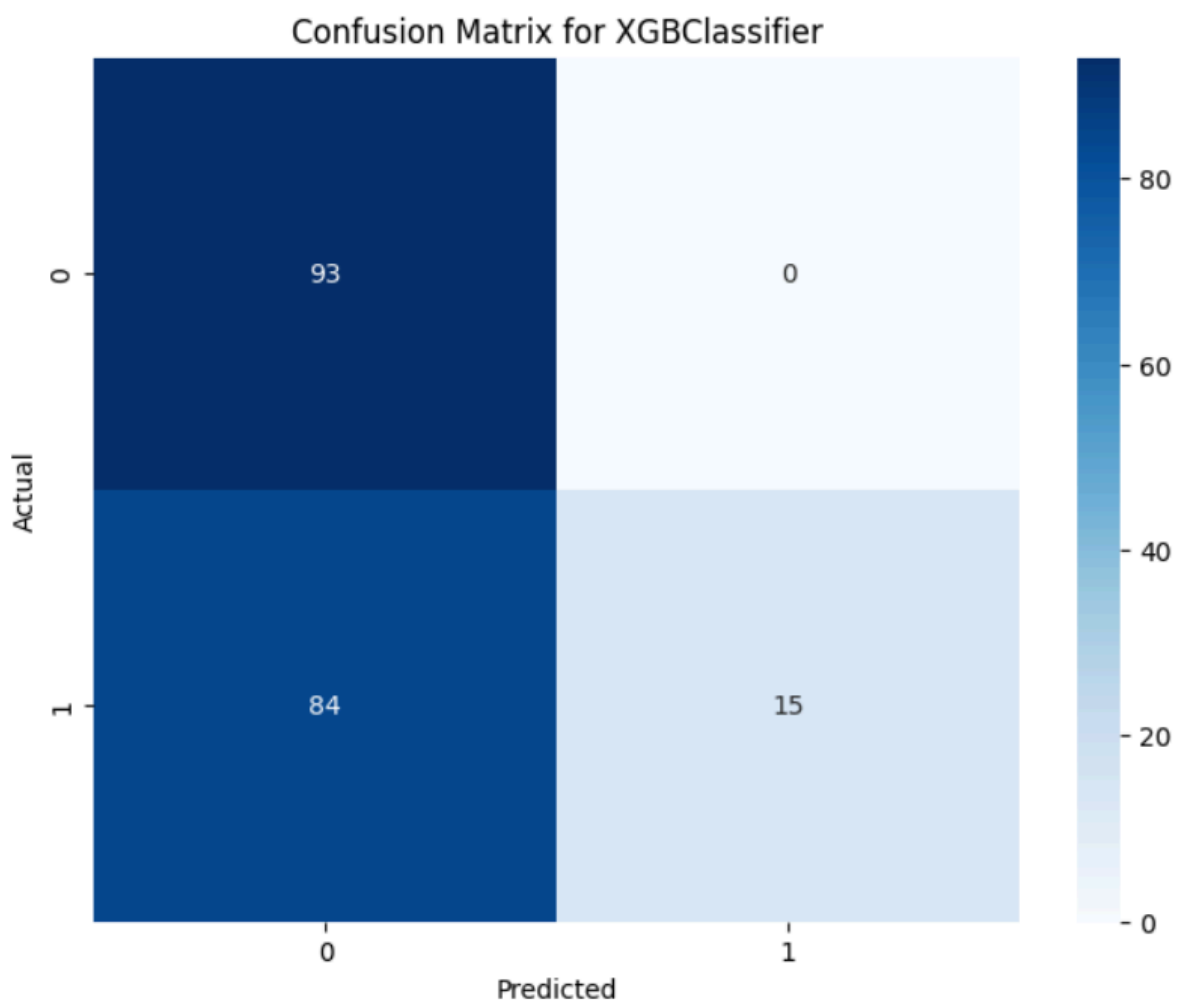


Рисунок 3.6 — Матриця плутанини моделі XGBoost

3.3.3 Порівняння результатів

Порівняння результатів двох моделей показує, що модель Random Forest є більш ефективною у виявленні атак типу Rapid Reset на нових даних у порівнянні з моделлю XGBoost. Модель Random Forest демонструє високу точність, повноту та F1-міру, а також швидкий час передбачення, що є важливим для використання в реальному часі.

Таблиця 3.5 Порівняння результатів

Модель	Precision (0)	Recall (0)	F1-score (0)	Precision (1)	Recall (1)	F1-score (1)	Accuracy	Prediction Time (s)
Random Forest	0.89	0.97	0.93	0.97	0.89	0.94	0.94	0.0075
XGBoost	0.53	0.97	0.69	0.97	0.15	0.26	0.56	0.0010

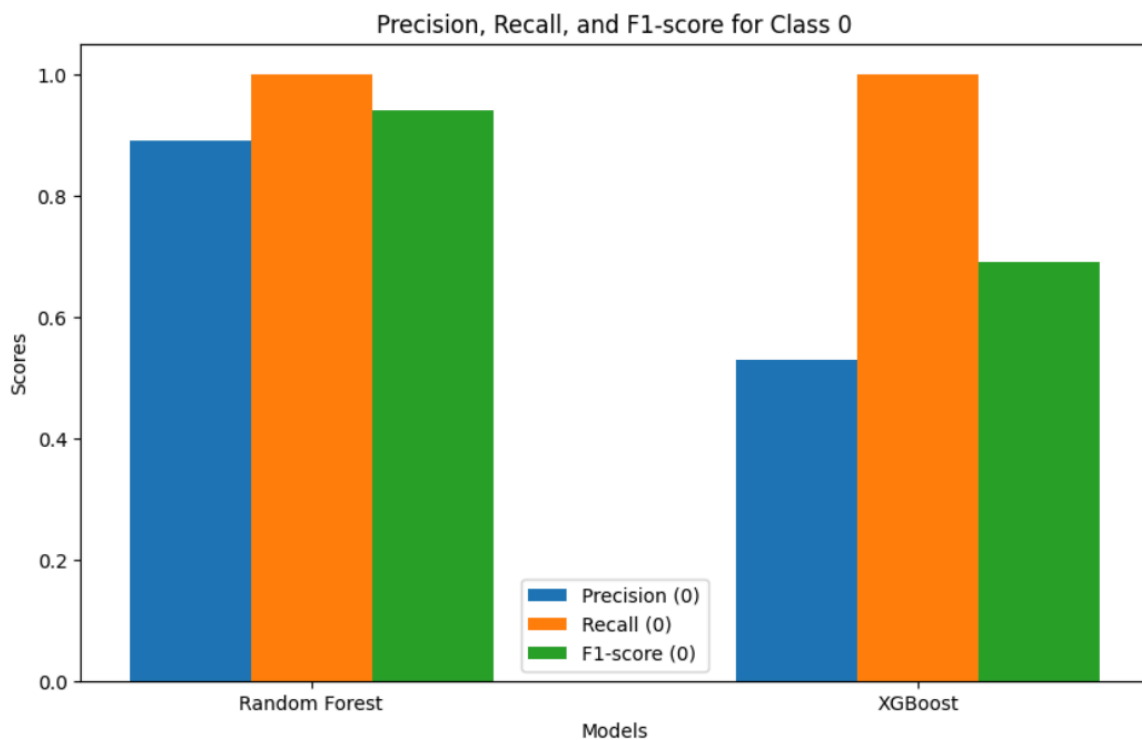


Рисунок 3.7 — Результати для класу 0

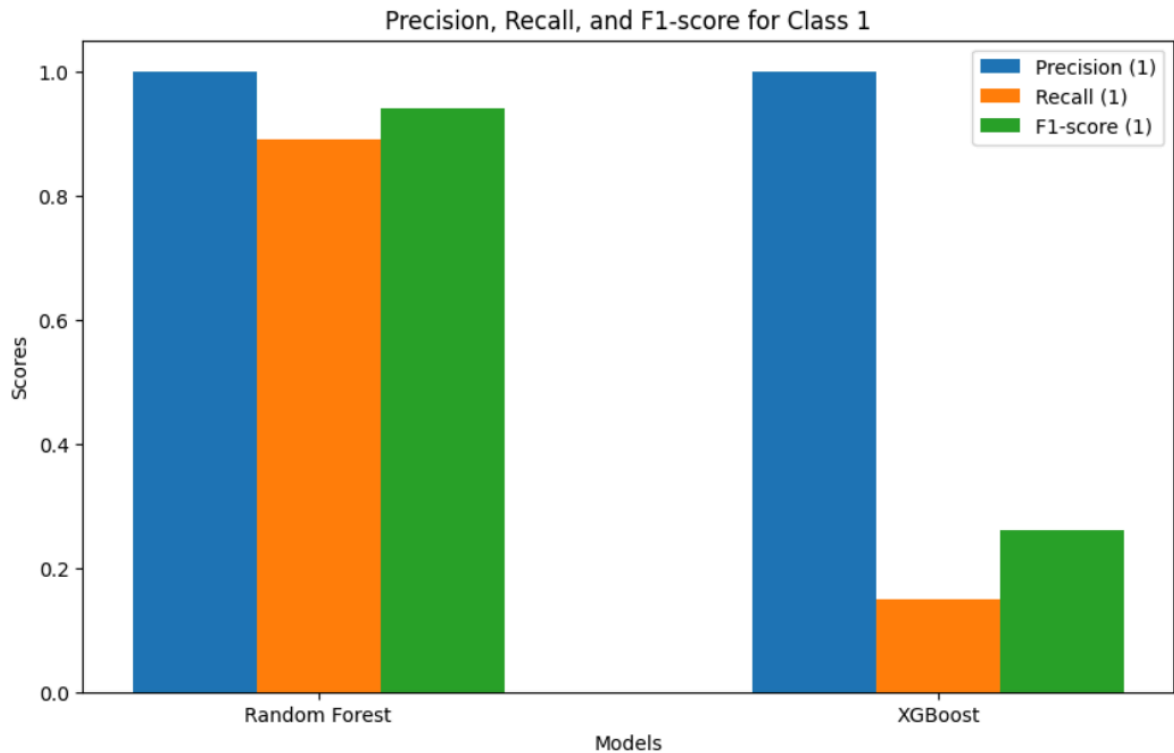


Рисунок 3.8 — Результати для класу 1

Висновки:

- Модель Random Forest показала високу точність і стабільність у виявленні атак типу Rapid Reset, що робить її оптимальним вибором для розгортання у реальних умовах.
- Модель XGBoost продемонструвала нижчу точність і ефективність, що вказує на необхідність подальшого налаштування або вибору інших моделей для покращення результатів.

ВИСНОВКИ З РОЗДІЛУ 3

У третьому розділі роботи було проведено комплексний аналіз навчання та тестування моделей для виявлення атак типу Rapid Reset. Для забезпечення якісного навчання моделей був проведений ретельний процес обробки даних. Зібрані дані трафіку були перетворені у формат CSV, що дозволило їх подальше використання у машинному навчанні. Було застосовано методи для заповнення пропущених значень, обробки тимчасових залежностей та балансування класів. Це забезпечило рівномірний розподіл даних між нормальним трафіком і атаками, що є критично важливим для побудови надійних моделей.

Були обрані дві моделі для навчання: Random Forest і XGBoost. Ці моделі були обрані через їхню здатність ефективно працювати з незбалансованими даними і надавати високу точність. Під час навчання було використано різні параметри для оптимізації моделей, а також метод SMOTE для подальшого балансування класів.

Тестування моделей було проведено на окремому наборі даних, щоб оцінити їхню здатність виявляти атаки Rapid Reset. Результати тестування для моделі XGBoost показали високу точність (0.97) у виявленні нормального трафіку, але помірну ефективність (0.15) у виявленні атак Rapid Reset. Модель Random Forest досягла точності (0.94) з високими значеннями precision та recall для обох класів: для класу 0 (нормальний трафік) precision склав 0.89, recall – 0.97, для класу 1 (атака) precision склав 0.97, recall – 0.89. Загальна точність моделі Random Forest склала 0.94.

Аналіз результатів показав, що обидві моделі добре справляються з класифікацією нормального трафіку, але стикаються з труднощами у виявленні атак. Це підкреслює необхідність додаткових досліджень і оптимізацій для поліпшення здатності моделей розпізнавати специфічні атаки, такі як Rapid Reset.

У результаті дослідження були досягнуті значні успіхи у розробці методів виявлення Rapid Reset атак за допомогою машинного навчання. Незважаючи на те, що існують області для покращення, особливо у виявленні атак, проведена робота забезпечує міцну основу для подальших досліджень і розвитку в цій сфері. Ефективність моделей Random Forest і XGBoost виявилася перспективною, що робить їх основними кандидатами для подальших удосконалень і тестувань.

ВИСНОВКИ

У цій роботі було досліджено кілька класифікацій DDoS-атак з особливою увагою до Rapid Reset атак, які використовують особливості протоколу HTTP/2 для створення великих навантажень на сервери через швидке скидання з'єднань. Розглянуто недоліки існуючих рішень захисту від Rapid Reset атак, включаючи обмеження кількості потоків та моніторинг кадрів RST_STREAM, які мають обмеження і не завжди забезпечують належний рівень захисту.

Враховуючи переваги нейронних мереж у виявленні аномалій, адаптивності, швидкій обробці великих обсягів даних та зниженні кількості помилкових спрацьовувань, було зроблено висновок про актуальність використання нейронних мереж для захисту від Rapid Reset атак.

У другому розділі детально розглянуто етапи підготовки та обробки даних для навчання нейронної мережі з метою виявлення Rapid Reset атак. Налаштування оточення включало підняття Docker-контейнера з веб-сервером на базі Flask та налаштування Nginx для підтримки HTTP/2, що дозволило генерувати та зібрати необхідний трафік. Процес симуляції трафіку забезпечував створення як нормального, так і атакуючого трафіку, який був зібраний за допомогою Wireshark. Дані були ретельно очищені та перетворені для аналізу, включаючи обробку тимчасових міток, заповнення відсутніх значень і кодування категоріальних змінних.

Були обрані моделі Random Forest і XGBoost, які добре справляються з незбалансованими даними і забезпечують високу точність. Процес навчання включав оптимізацію параметрів та використання методу SMOTE для подальшого балансування класів.

У третьому розділі проведено комплексний аналіз навчання та тестування моделей. Модель XGBoost показала високу точність у виявленні нормального трафіку, але помірну ефективність у виявленні атак Rapid Reset. Модель Random Forest досягла загальної точності 0.94 з високими значеннями precision та recall для обох класів.

Аналіз результатів показав, що обидві моделі добре справляються з класифікацією нормального трафіку, але стикаються з труднощами у виявленні атак. Це підкреслює необхідність додаткових досліджень і оптимізацій для поліпшення здатності моделей розпізнавати специфічні атаки, такі як Rapid Reset.

В результаті дослідження були досягнуті значні успіхи у розробці методів виявлення Rapid Reset атак за допомогою машинного навчання. Моделі Random Forest і XGBoost показали перспективні результати, що робить їх основними кандидатами для подальших удосконалень і тестувань.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. Cloudflare, Ping (ICMP) flood DDoS attack – 2021 –
<https://www.cloudflare.com/learning/ddos/ping-icmp-flood-ddos-attack/>
2. Cloudflare, How do layer 3 DDoS attacks work? | L3 DDoS – 2021 –
<https://www.cloudflare.com/learning/ddos/layer-3-ddos-attacks/>
3. Cloudflare, UDP flood attack – 2021 –
<https://www.cloudflare.com/learning/ddos/udp-flood-ddos-attack/>
4. Cloudflare, SYN flood attack – 2021 –
<https://www.cloudflare.com/learning/ddos/syn-flood-ddos-attack/>
5. Cloudflare, HTTP flood attack – 2021 –
<https://www.cloudflare.com/learning/ddos/http-flood-ddos-attack/>
6. Cloudflare, Slowloris DDoS attack – 2021 –
<https://www.cloudflare.com/learning/ddos/ddos-attack-tools/slowloris/>
7. Countering DNS Amplification Attacks Based on Analysis of Outgoing Traffic – 2023 – <https://ieeexplore.ieee.org/abstract/document/10173727>
8. Cloudflare, NTP amplification DDoS attack – 2021 –
<https://www.cloudflare.com/ru-ru/learning/ddos/ntp-amplification-ddos-attack/>
9. HTTP/2 Rapid Reset: deconstructing the record-breaking attack – 2023 –
<https://blog.cloudflare.com/technical-breakdown-http2-rapid-reset-ddos-attack>
10. Rapid Reset DDoS: How It Works and How Gcore Protects Customers – 2023 – <https://gcore.com/news/rapid-reset-ddos/>
11. Machine Learning Techniques to Detect a DDoS Attack in SDN: A Systematic Review – 2023 – <https://www.mdpi.com/2076-3417/13/5/3183>

12. A general architecture of Random Forest – 2019 –

https://www.researchgate.net/figure/A-general-architecture-of-Random-Forest-5_fig2_335483097

13. A general architecture of XGBoost – 2019 –

https://www.researchgate.net/figure/A-general-architecture-of-XGBoost_fig3_335483097