ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА КОМП'ЮТЕРНИХ НАУК

КВАЛІФІКАЦІЙНА РОБОТА

на тему: «Додаток на мові Python,для аналізу Dos,DDos атак»

на здобуття освітнього ступеня бакалавра
зі спеціальності «122 Комп'ютерні науки»
(код, найменування спеціальності)
освітньо-професійної програми Комп'ютерні науки
(назва)

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело

_____ Артем ЩЕРБА

(підпис) *Ім'я, ПРІЗВИЩЕ здобувача*

Виконав: здобувач(ка) вищої освіти гр. КНД-42
Артем ЩЕРБА
Ім'я, ПРІЗВИЩЕ
Керівник: Юлія БЕРЕЗОВСЬКА
Науковий ступінь, вчене звання Ім'я, ПРІЗВИЩЕ
Рецензент: _____
науковий ступінь, вчене звання Ім'я, ПРІЗВИЩЕ

Київ 2025

ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

Навчально-науковий інститут Інформаційних технологій

Кафедра Комп'ютерних наук

Ступінь вищої освіти Бакалавр

Спеціальність 122 Комп'ютерні науки

Освітньо-професійна програма Комп'ютерні науки

ЗАТВЕРДЖУЮ

Завідувач кафедрою Комп'ютерних наук

Віктор ВИШНІВСЬКИЙ

Ім'я, ПРІЗВИЩЕ

«____» _____ 20__ р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Щерба Артем Андрійович

(прізвище, ім'я, по батькові здобувача)

1.Тема кваліфікаційної роботи: "Додаток на мові Python,для аналізу Dos,DDos атак"

керівник кваліфікаційної роботи Юлія БЕРЕЗОВСЬКА, доцент кафедри, доктор філософії ,

(Ім'я, ПРІЗВИЩЕ, науковий ступінь, вчене звання)

затверджені наказом Державного університету інформаційно-комунікаційних технологій від «24» лютого 2025 р. №56

2. Строк подання кваліфікаційної роботи «31» травня 2025р.

3.Вихідні дані до кваліфікаційної роботи: Науково-технічна література з питань кібербезпеки, аналізу мережевого трафіку, протоколів TCP/UDP, а також документація до бібліотек Python (Scapy, PyQt5, Matplotlib, Folium).

4.Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)

4.1 Аналіз потреб у виявленні DoS/DDoS-атак, постановка завдань розробки програмного забезпечення.

4.2Фахівці з кібербезпеки, мережеві адміністратори, аналітики інформаційної безпеки.

4.3 Мова програмування Python, бібліотеки Scapy, Matplotlib, PyQt5, Folium; застосування API ip-api та ipwho.is; побудова GUI через Qt.
5.Перелік ілюстративного матеріалу: презентація
1.Схема архітектури додатку
2.Фрагменти коду
3.Знімки екрану GUI
4.Графіки розподілу атак
5. Інтерактивна карта IP-джерел.
6. Дата видачі завдання«24» лютого 2025 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Підбір науково-технічної літератури	25.02-27.02	Виконано
2	Дослідження поставленої задачі	27.02-02.03	Виконано
3	Аналіз методів розв’язування задачі	03.03-07.03	Виконано
4	Застосування застосунку на практиці	09.03-14.03	Виконано
5	Аналіз отриманих результатів	16.03-21.03	Виконано
6	Розробка прототипу	23.03-20.04	Виконано
7	Вступ, висновки, реферат	24.04-09.05	Виконано
8	Основні розділи	12.05-19.05	Виконано
9	Попередній захист роботи	26.05-28.05	Виконано

Здобувач(ка) вищої освіти _____ ___Артем ЩЕРБА___

(підпис) *(Ім’я, ПРІЗВИЩЕ)*

Керівник

кваліфікаційної роботи _____ ___Юлія БЕРЕЗОВСЬКА_____

(підпис) *(Ім’я, ПРІЗВИЩЕ)*

РЕФЕРАТ

Текстова частина кваліфікаційної роботи на здобуття освітнього ступеня бакалавра: 50 стор., 15 рис., 20 джерел.

Мета роботи:-розробка та реалізація програмного додатку на мові Python для аналізу мережевого трафіку з метою виявлення DoS та DDoS атак.

Об’єкт дослідження:-мережевий трафік, що передається через комп’ютерні мережі.

Предмет дослідження:-методи аналізу мережевого трафіку для виявлення ознак DoS та DDoS атак.

Методи дослідження:

- Пакетний аналіз даних з використанням бібліотеки Scapy;
- Геолокаційне визначення країни IP-адрес через відкриті API (ip-api, ipwho.is);
- Візуалізація даних з використанням бібліотек Matplotlib та Folium;
- Методи статистичного аналізу для виявлення аномалій у частоті запитів;
- Проєктування та реалізація графічного інтерфейсу з використанням PyQt5.

Сфера застосування:

- Інформаційна безпека підприємств
- Адміністрування комп’ютерних мереж
- Освітні та дослідницькі цілі у сфері кібербезпеки
- Підтримка систем моніторингу мережевого трафік

ЗМІСТ

ВСТУП

Сучасний етап розвитку інформаційних технологій характеризується активною цифровізацією усіх сфер суспільного життя. Від електронного урядування і банківських транзакцій до функціонування транспортної та енергетичної інфраструктури — усе дедалі більше залежить від надійності інформаційних систем. Проте разом із розширенням цифрового простору стрімко зростає й спектр потенційних кіберзагроз. Одними з найнебезпечніших серед них залишаються атаки типу відмова в обслуговуванні (DoS) та розподілена відмова в обслуговуванні (DDoS), що є ефективним інструментом кіберзлочинців для виведення з ладу критичних об’єктів IT-інфраструктури.

Сутність таких атак полягає в навмисному перевантаженні цільового ресурсу (сервера, веб-сайту або мережі) великою кількістю запитів, що унеможливлює обробку легітимного трафіку. DDoS-атаки реалізуються зазвичай за допомогою великої кількості скомпрометованих пристроїв (ботнетів), що значно ускладнює їх виявлення та нейтралізацію. Складність полягає ще й у тому, що трафік від ботів на перший погляд може не відрізнятися від нормального. За останні роки методи проведення атак значно еволюціонували — від примітивних SYN flood до складних мультивекторних атак, здатних адаптуватися до захисних механізмів у режимі реального часу [1, с. 108].

У цьому контексті особливо актуальним є створення інструментів аналізу мережевого трафіку, що дозволяють виявляти аномалії до того, як вони призведуть до значних втрат. Замість простого реагування на атаку вже після її початку, дедалі більше компаній прагнуть до проактивного підходу: виявлення підозрілої активності, побудови профілів звичайного трафіку та фіксації відхилень.

Метою цієї дипломної роботи є розробка програмного додатку для виявлення DoS та DDoS-атак шляхом аналізу PCAP-файлів, які містять захоплений мережевий трафік.Додаток дозволяє виконувати попередню обробку даних

виявляти ключові ознаки атак, класифікувати активність за протоколами, а також візуалізувати отриману інформацію. Окрему увагу приділено побудові зручного графічного інтерфейсу, який дозволяє спеціалісту швидко інтерпретувати результати без глибокого занурення в програмний код.

Реалізація інструменту здійснена мовою програмування Python, яка надає широкий спектр бібліотек для обробки трафіку, побудови графіки та GUI. Зокрема, у розробці використано:

- Scapy — для парсингу мережевих пакетів;
- Matplotlib — для побудови статистичних графіків;
- Folium — для візуалізації IP-адрес на інтерактивній карті;
- PyQt5 — для створення багатовкладкового графічного інтерфейсу користувача.

Дослідження, отримані під час роботи над додатком, частково були представлені автором у тезах до студентської наукової конференції, де акцентовано практичне значення та потенціал розробки [1, с. 109].

Таким чином, дипломна робота поєднує в собі теоретичний аналіз сучасних підходів до виявлення DoS/DDoS-атак та практичну реалізацію програмного засобу, що має прикладне значення для фахівців у галузі кібербезпеки, системних адміністраторів та розробників систем захисту інформації.

1 Аналітичний огляд методів виявлення DoS та DDoS атак

Атаки типу Denial of Service (DoS) та Distributed Denial of Service (DDoS) залишаються одними з найнебезпечніших загроз у сфері кібербезпеки. Їх основна мета — порушити доступність сервісів, шляхом перевантаження мережевих ресурсів великою кількістю запитів або пакетів. З кожним роком масштаби цих атак зростають, ускладнюючи роботу традиційних засобів захисту. У зв'язку з цим виникає потреба в ефективних методах виявлення, здатних ідентифікувати аномальні шаблони трафіку та забезпечити своєчасне реагування на загрози.

1.1 Класифікація методів виявлення

Методи виявлення DoS/DDoS атак умовно поділяються на декілька основних підходів, кожен із яких має свої переваги та обмеження:

Методи на основі правил (Rule-based): використовують фіксовані сигнатури або шаблони відомих атак. При надходженні трафіку система порівнює його характеристики з базою відомих атак. Перевагою є швидкість і простота реалізації, але основним недоліком є нездатність виявляти нові або модифіковані атаки, які не відповідають жодному з наявних шаблонів [3, с. 82].

Статистичні методи (Anomaly-based): базуються на встановленні базових рівнів нормальної активності мережі та виявленні відхилень. Такі методи дозволяють виявляти незнайомі або змінені атаки, однак можуть генерувати велику кількість хибних спрацювань, особливо у випадках різких змін у легітимному трафіку [4, с. 114].

Методи машинного навчання: аналізують вхідні дані з використанням алгоритмів класифікації (Decision Trees, SVM, Random Forest), кластеризації (K-Means) або глибокого навчання (нейронні мережі). Їхня перевага — здатність навчатися на основі реального трафіку, що дозволяє виявляти як відомі, так і нові вектори атак. Але їх недоліками є потреба у великих обсягах якісних даних для навчання та високі обчислювальні витрати [5, с. 35].

Гібридні методи: поєднують кілька підходів — наприклад, сигнатурний аналіз і статистичний моніторинг — що дозволяє компенсувати недоліки окремих методів та підвищити точність виявлення.

1.2 Ключові ознаки DDoS трафіку

Ефективне виявлення атак типу Distributed Denial of Service (DDoS) передбачає здатність системи виявити аномалії у поведінці мережевого трафіку. Такі атаки не супроводжуються зламом або проникненням у систему, але призводять до повного або часткового порушення її працездатності. Важливо розуміти типові ознаки, за якими шкідливий трафік можна відрізнити від легітимного.

Однією з найочевидніших ознак DDoS-атак є аномально висока частота однотипних мережевих запитів. У випадку SYN flood-атаки фіксується велика кількість TCP SYN-пакетів, які не супроводжуються завершенням з'єднання (відсутні ACK-пакети). Це призводить до перевантаження таблиці з'єднань на сервері. Часто кількість таких запитів сягає десятків або навіть сотень тисяч на секунду [6, с. 56].

Ще однією ключовою особливістю є розподілений характер атак. На відміну від традиційного DoS, у якому джерело трафіку — одна машина, у DDoS-атаках використовується ботнет — мережа скомпрометованих пристроїв, які генерують запити з різних IP-адрес одночасно. Це унеможливлює просте блокування за IP або маскою підмережі, адже запити надходять нібито від «різних» користувачів.

Розмір пакетів також може бути індикатором аномалії. Наприклад, атаки типу DNS Amplification використовують спеціально сформовані запити, які, будучи віддзеркаленими від відкритих DNS-серверів, створюють у відповідь пакети набагато більшого розміру. У типовому випадку можна спостерігати пакети розміром понад 300 байтів, які надходять із великою частотою.

Також високий рівень невдалих спроб встановлення TCP-з’єднання — наприклад, SYN-пакети, які не завершуються ACK-підтвердженнями — свідчить про можливе використання тактики «виснаження ресурсів» сервера.

Ознаки ICMP-flood атак включають масову генерацію ICMP Echo-пакетів (ping-запитів), часто з підробленими джерелами, що також призводить до перевантаження ресурсів, зокрема вхідних каналів або обчислювальної потужності.

У межах розробленого додатку реалізовано автоматизований аналіз зазначених характеристик із використанням бібліотеки Scapy. Цей інструмент дає змогу зчитувати PCAP-файли, ідентифікувати тип кожного пакета та відстежувати кількість SYN-, UDP-, ICMP-пакетів у розрізі IP-адрес. На основі зібраної інформації формується таблиця активності, а також візуалізація трафіку, що дає змогу оперативно виявити підозрілі шаблони.

Таким чином, правильне розуміння ознак DDoS-атак та їхня фіксація в системі дозволяє не лише аналізувати наслідки інцидентів, але й розробляти засоби раннього попередження та реагування.

1.3 Інструменти для аналізу

Існує широкий спектр інструментів, які активно застосовуються у сфері аналізу мережевого трафіку та виявлення атак типу DoS/DDoS. Вибір конкретного інструменту зазвичай залежить від цілей дослідження, необхідного рівня автоматизації, глибини аналізу, а також технічних можливостей аналітика або системи. Нижче наведено огляд найпоширеніших рішень.

Wireshark є одним із найпотужніших і найпопулярніших інструментів для графічного аналізу мережевого трафіку. Він дозволяє у реальному часі перехоплювати та досліджувати пакети різних протоколів, зокрема TCP, UDP, ICMP, HTTP тощо. Серед його основних переваг – зручний інтерфейс, підтримка фільтрів, статистики та розширеного декодування пакетів. Проте варто зазначити, що Wireshark переважно орієнтований на ручну роботу спеціаліста, що обмежує

його застосування в задачах автоматизованого моніторингу або великомасштабного аналізу [7, с. 29].

Snort та Suricata — це високопродуктивні системи виявлення та запобігання вторгненням (IDS/IPS), які використовують сигнатурні правила для виявлення шкідливої активності. Snort має довгу історію розвитку і підтримує велику базу правил, що дозволяє ефективно розпізнавати відомі шаблони атак. Suricata, у свою чергу, є більш сучасною системою, яка також підтримує багатопоточність і працює з PCAP-файлами. Обидві системи можуть бути інтегровані у більші комплекси захисту інформаційної інфраструктури.

Bro (нині Zeek) — це потужна платформа для детального аналізу мережевого трафіку. Вона надає можливість створення власних сценаріїв обробки подій за допомогою спеціальної мови програмування, що дозволяє користувачу гнучко адаптувати поведінку системи до специфіки власної мережі. Zeek більше орієнтований на контекстну обробку даних, ніж на сигнатурне виявлення.

Окремо варто відзначити Python-інструменти, які стали надзвичайно популярними серед дослідників та розробників завдяки своїй відкритості, гнучкості та інтеграційним можливостям. Такі бібліотеки, як Scapy, Pandas, Matplotlib дозволяють створювати кастомізовані рішення для аналізу трафіку, візуалізації активності, побудови графіків, гістограм, аналізу часових рядів та геолокації IP-адрес. Зокрема, Scapy дає змогу зчитувати PCAP-файли, фільтрувати пакети, ідентифікувати підозрілі шаблони (наприклад, SYN flood або DNS Amplification), тоді як Pandas дозволяє структурувати дані, а Matplotlib – ефективно їх візуалізувати.

Саме Python-підхід обрано у розробленому нами застосунку, оскільки він забезпечує гнучкість, можливість розширення функціоналу, адаптацію під конкретні сценарії та простоту розгортання. На відміну від готових рішень з фіксованими правилами, Python-реалізація дозволяє комбінувати кілька методів аналізу, інтегрувати API та реалізовувати кастомну логіку обробки трафіку, що є критично важливим для аналізу сучасних DDoS-атак, які постійно еволюціонують.

1.4 Висновок до розділу 1

У результаті аналітичного огляду було визначено основні підходи до виявлення DoS та DDoS атак, серед яких виділяються сигнатурні, статистичні, методи машинного навчання та гібридні технології. Окрему увагу приділено типовим ознакам DDoS-атак та сучасним інструментам аналізу мережевого трафіку, що стали основою для вибору методології у практичній частині дослідження.

2 Обґрунтування вибору інструментів та технологій

У процесі створення програмного додатку для аналізу DoS та DDoS атак було обрано низку інструментів, які є не лише сучасними, а й широко використовуються в практиці кібербезпеки. Вибір базувався на низці технічних та практичних критеріїв, зокрема: відкритість програмного забезпечення, сумісність із мовою програмування Python, наявність активної спільноти розробників, підтримка документації, можливість інтеграції з іншими бібліотеками, а також стабільна робота в умовах обробки великих обсягів мережевого трафіку [1, с. 108].

Мова програмування Python стала основою реалізації завдяки своїй гнучкості, простому синтаксису та великому вибору бібліотек, які дозволяють реалізовувати як обробку мережевого трафіку, так і його візуалізацію, створення графічного інтерфейсу користувача та взаємодію з API. Особливу роль зіграла підтримка мультиплатформенності, що забезпечує можливість запуску програми як на Windows, так і на Linux-системах без суттєвих змін у коді.

Для аналізу мережевого трафіку використано бібліотеку Scapy, яка дозволяє зчитувати PCAP-файли, фільтрувати пакети за протоколами (TCP, UDP, ICMP), аналізувати структуру пакетів та виявляти шаблони підозрілої активності (наприклад, SYN-flood або DNS-ампліфікації). Scapy добре зарекомендувала себе в сфері розробки IDS/IPS систем і є основним інструментом аналізу трафіку у програмі [2, с. 223].

Графічний інтерфейс було реалізовано з використанням PyQt5 — фреймворку, який дозволяє створювати багатовіконні інтерфейси з вкладками, таблицями, кнопками, прогрес-барами та іншими елементами керування. PyQt5 забезпечив зручність у візуальному представленні інформації та підтримку реактивної взаємодії з користувачем.

Візуалізація результатів аналізу трафіку здійснюється через Matplotlib (для побудови графіків, діаграм активності) та Folium (для створення інтерактивної карти IP-адрес з геолокацією). Візуальна складова значно покращує сприйняття аналітичної інформації, особливо в контексті визначення країн-джерел потенційних атак [6, с. 64; 7, с. 203].

Для визначення географічного розташування IP-адрес реалізовано каскадну систему запитів до зовнішніх API: у першу чергу використовується сервіс ip-api.com, а при відмові — резервний ipwho.is. Це дозволяє зменшити кількість невизначених адрес та отримати точні координати атакуючих вузлів [8, с. 158].

Завдяки використанню вищезгаданих інструментів вдалося створити модульну систему, яка поєднує засоби збору, обробки та представлення даних у єдиній програмній платформі. Такий підхід забезпечує не лише зручність користування, а й відкриває можливості для масштабування системи в майбутньому — зокрема, для інтеграції з хмарними сервісами, реалізації онлайн-моніторингу або підключення до SIEM-рішень.

2.1 Вибір мови програмування Python

У процесі розробки програмного забезпечення для аналізу DoS та DDoS атак ключовим рішенням стало використання мови програмування Python, яка на сьогодні є однією з найпоширеніших та найбільш гнучких мов у сфері кібербезпеки, аналізу даних та побудови прототипів систем реального часу. Вибір Python був обумовлений як технічними, так і практичними факторами, що забезпечили ефективність реалізації всіх компонентів системи в єдиному середовищі [1, с. 108].

Python підтримує розробку як прикладних, так і системних рішень, зокрема завдяки модульності та багатій екосистемі бібліотек. Це дозволило реалізувати в межах одного проєкту такі функціональні блоки, як:

- захоплення та обробка мережевого трафіку з використанням Scapy;
- візуалізація даних через Matplotlib і Folium;
- побудова графічного інтерфейсу користувача з PyQt5;
- робота з зовнішніми API для геолокації IP-адрес;
- логічна обробка даних у реальному або майже реальному часі.

Серед ключових переваг мови Python, які мали вирішальне значення при розробці додатку, можна виділити:

- Зрозумілий та лаконічний синтаксис, який значно спрощує підтримку коду, підвищує читабельність та зменшує кількість помилок на етапі розробки;
- Велика кількість готових бібліотек для роботи з мережею, візуалізацією, геолокацією, GUI, що дозволяє уникати "винаходу велосипеда" та зосередитися на реалізації логіки проєкту;
- Кросплатформеність, яка дозволяє запускати програму на Windows, Linux або macOS без істотних змін у коді;
- Активна спільнота, що забезпечує швидке знаходження рішень у випадку виникнення помилок або необхідності в оптимізації;
- Простота інтеграції з іншими технологіями, включно з C/C++, Java, SQL, Web API тощо.

Окремо варто зазначити, що Python ідеально підходить для обробки PCAP-файлів, особливо у поєднанні з бібліотекою Scapy. Завдяки цьому стало можливим реалізувати ефективний механізм зчитування, фільтрації, аналізу та класифікації великої кількості мережевих пакетів.

Використання Python також спрощує процес тестування та розширення функціоналу. Наприклад, за необхідності можна швидко додати нові модулі, інтегрувати логування, реалізувати онлайн-моніторинг або підключити систему сповіщень.

Таким чином, Python не лише забезпечив необхідний рівень продуктивності та гнучкості для розробки функціонального додатку, а й дозволив створити розширювану платформу, придатну для подальшого розвитку, включно з інтеграцією в більш складні системи моніторингу та кіберзахисту.

2.2 Бібліотека Scapy

Однією з ключових технологій, використаних у реалізації додатку для аналізу DoS та DDoS атак, є бібліотека Scapy — потужний інструмент для створення, надсилання, захоплення, маніпуляції та аналізу мережевих пакетів на різних рівнях моделі OSI. Scapy реалізована на мові Python і дозволяє розробникам повністю контролювати структуру мережевого трафіку, що робить її ідеальним вибором для задач кібербезпеки, тестування мереж та дослідження мережевих протоколів [2, с. 223].

У межах проєкту Scapy виконує центральну роль у модулі аналізу. Зокрема, вона використовується для:

- Зчитування PCAP-файлів, які є стандартом для збереження мережевих трасувань;
- Ідентифікації ключових протоколів, таких як TCP, UDP, ICMP, HTTP, що є критично важливими при класифікації типів трафіку;
- Аналізу ознак атак, таких як:
- велика кількість SYN-пакетів без відповіді — типовий шаблон SYN flood атак;
- наявність UDP-пакетів з великим розміром корисного навантаження, що характерно для DNS Amplification атак;
- підозрілі ICMP-пінги у великих обсягах — ознака ICMP flood атак.

Окрім того, Scapy дозволяє гнучко фільтрувати пакети, що пришвидшує процес аналізу. Наприклад, за допомогою простих умовних виразів можна обробляти лише пакети з конкретними ознаками (наприклад, лише TCP SYN або лише ICMP). Це дозволяє значно зменшити обсяг оброблюваної інформації та зосередитися на найбільш критичних аспектах мережевого трафіку.

Ще однією сильною стороною Scapy є інтеграція з іншими бібліотеками. Наприклад, результати аналізу, зібрані за допомогою Scapy, можуть бути легко

передані у модуль візуалізації (Matplotlib або Folium), або використані у графічному інтерфейсі користувача, побудованому на PyQt5. Завдяки відкритій архітектурі та підтримці розширень, Scapy може бути адаптована під нові протоколи або специфічні сценарії виявлення аномалій у трафіку.

Scapy також підтримує розширений доступ до полів заголовків пакетів, що дозволяє виявляти навіть нестандартні шаблони атак, які не були передбачені у типових сигнатурах. Така гнучкість дозволяє використовувати бібліотеку не лише для виявлення відомих атак, але й для дослідження нових потенційних загроз.

Загалом, вибір Scapy обумовлений її продуктивністю, універсальністю та орієнтованістю на низькорівневу обробку мережевого трафіку, що є надзвичайно важливим у контексті аналізу DoS/DDoS атак. Ця бібліотека надала основу для реалізації основної логіки обробки даних у програмі, і, при потребі, може бути розширена для підтримки нових векторів атак або нестандартних типів мережевого трафіку.

2.3 PyQt5 для побудови GUI

Для розробки графічного інтерфейсу користувача у створеному додатку було обрано бібліотеку PyQt5, яка є популярним інструментом для створення сучасних кросплатформних інтерфейсів на мові Python. PyQt5 поєднує потужність фреймворку Qt із простотою Python, що робить її зручною для побудови інтерактивних вікон, вкладок, діалогів та інших елементів керування [5, с. 77].

Основними перевагами використання PyQt5 є:

- Можливість створення багатовіконних інтерфейсів, які легко масштабуються та адаптуються до потреб користувача;
- Підтримка великої кількості віджетів, серед яких: кнопки, таблиці, текстові поля, прогрес-бари, випадаючі списки тощо;
- Інтеграція з іншими бібліотеками, такими як Matplotlib (для графіків) та QtWebEngine (для відображення HTML-контенту, зокрема інтерактивної карти Folium);

- Наявність механізму сигналів та слотів, який забезпечує ефективну взаємодію між частинами додатку (наприклад, запуск аналізу після натискання кнопки або оновлення прогрес-бара).
- У межах нашого додатку інтерфейс побудовано у вигляді вкладок:
- Traffic Table — таблиця, де виводяться всі виявлені IP-адреси, кількість підозрілих пакетів за протоколами, часові мітки першого та останнього пакету, а також країна походження.
- IP Map — інтерактивна карта, згенерована через бібліотеку Folium, яка відображається у вікні за допомогою QtWebEngineView. Кожен маркер на карті відповідає IP-адресі, отриманій у процесі аналізу.
- Country Chart — гістограма, побудована на основі Matplotlib, яка ілюструє кількість IP-адрес за країнами, дозволяючи швидко ідентифікувати джерела атаки.

Крім вкладок, інтерфейс містить:

- Кнопки для завантаження PCAP-файлів;
- Прогрес-бар, який відображає стан виконання аналізу в реальному часі;
- Інформативне поле оцінки часу, що орієнтує користувача щодо тривалості обробки.

Дизайн витримано у темних тонах (dark mode), що знижує навантаження на очі та робить інтерфейс візуально привабливішим для користувача. Завдяки можливості стилізації елементів через CSS-подібні стилі (Qt Style Sheets), було реалізовано сучасний вигляд програми з акцентом на зручність і читабельність.

Таким чином, використання PyQt5 дозволило створити не лише функціональний, а й інтуїтивно зрозумілий інтерфейс, у якому навіть користувач без глибоких технічних знань може швидко завантажити файл, переглянути підозрілі IP-адреси та ознайомитися з географічною картиною мережевої атаки. Завдяки модульності PyQt5, інтерфейс у майбутньому легко доповнити новими функціями, такими як фільтрація даних, експортування в інші формати або інтеграція з хмарними сервісами моніторингу.

2.4 Matplotlib і Folium для візуалізації

Візуалізація є ключовим елементом при аналізі DoS та DDoS атак, оскільки вона дозволяє швидко та інтуїтивно оцінити масштаби аномалій у трафіку, виявити підозрілу активність та отримати уявлення про географічне розташування джерел атак. У розробленому додатку для реалізації візуального аналізу було обрано дві потужні бібліотеки: Matplotlib та Folium.

- Бібліотека Matplotlib використовується для побудови графіків та гістограм, які відображають:
- Розподіл мережевої активності по країнах – це дозволяє визначити, з яких регіонів надходить найбільше підозрілого трафіку.
- Частоту запитів за протоколами (TCP, UDP, ICMP) – допомагає встановити, який саме тип трафіку переважає в потенційній атаці.
- Розподіл подій у часі – зокрема, SYN-, UDP-, ICMP- та HTTP-пакетів, що дозволяє виявити пікові навантаження.

Графіки побудовані з використанням інтерактивного віджета FigureCanvas, який вбудовується в GUI додатку через PyQt5. Завдяки цьому користувач може переглядати оновлення графіків у режимі реального часу одразу після завершення аналізу. Візуальне представлення даних у вигляді діаграм значно спрощує оцінку ситуації порівняно з текстовими звітами або числовими таблицями [6, с. 64].

Додатково, для візуалізації географічного походження IP-адрес у додатку інтегровано бібліотеку Folium. Вона дозволяє створювати інтерактивні HTML-карти з використанням Leaflet.js. На карті автоматично розміщуються маркери у координатах, отриманих з геолокаційних API, з підписами, що містять IP-адресу та назву країни.

Карта генерується динамічно під час аналізу трафіку, зберігається у локальний HTML-файл, а потім відкривається у вкладці додатку за допомогою компонента QtWebEngineView, який забезпечує повну інтерактивність карти без потреби у сторонньому браузері. Це дає змогу:

- швидко оцінити географічну концентрацію атак;
- виявити підозрілі IP, що походять з однієї або кількох країн;
- використовувати карту як візуальний індикатор масштабності загроз [7, с. 203].

Поєднання графіків Matplotlib та карт Folium у рамках єдиного GUI дозволяє досягти високої наочності та ефективності під час роботи з великою кількістю мережевого трафіку. Це особливо важливо в контексті навчального застосування, аудиту безпеки, а також у процесі прийняття рішень у відповідь на загрози. Візуалізація стає не лише засобом подання результатів, а й інструментом глибшого аналізу та обґрунтованих дій аналітика безпеки.

2.5 Запити до зовнішніх API

Одним із важливих компонентів реалізованого додатку є визначення географічного розташування джерел мережевого трафіку. Цей процес здійснюється шляхом надсилання HTTP-запитів до зовнішніх геолокаційних API-сервісів, які, на основі IP-адрес, повертають інформацію про країну, місто, координати, а іноді й організацію, через яку передається трафік.

Для цього у проєкті було обрано два популярні публічні сервіси — ip-api.com та ipwho.is, які забезпечують доступ до геолокаційної інформації без обов’язкової реєстрації та ключів доступу, що робить їх зручними для інтеграції у навчальні або експериментальні проєкти.

Основні переваги вибраних сервісів:

- ip-api.com надає зручний формат відповіді у JSON, повертаючи такі параметри, як: країна, місто, широта та довгота, регіон, назва провайдера, статус запиту. Його швидкодія дозволяє отримувати відповіді в середньому за 200–500 мс, що цілком задовольняє потреби додатку у фоновому режимі роботи.
- ipwho.is виконує схожі функції і також повертає повну геолокаційну інформацію. Його перевагою є стабільна робота, чіткий формат відповіді та

додаткові параметри, які можуть бути використані для розширеного аналізу в майбутньому (наприклад, часова зона, мова, валюта тощо).

- У розробленому додатку реалізовано каскадну логіку використання API. Це означає, що первинний запит відправляється на ip-api.com, і лише у випадку недоступності цього сервісу або некоректної відповіді (наприклад, статус "fail" або таймаут), виконується запит до резервного API — ipwho.is. Такий підхід дозволяє мінімізувати кількість втрат даних і забезпечити надійність при геолокації великої кількості IP-адрес.

Реалізація відбувається у фоновому потоці, що не блокує основний інтерфейс користувача. Результати зберігаються у кеші (словнику), що дозволяє не повторювати запити для одних і тих самих IP-адрес у межах однієї сесії аналізу.

Завдяки обраному підходу вдалось досягти балансу між продуктивністю, точністю та надійністю, що особливо важливо в задачах попереднього аналізу DoS/DDoS трафіку. Хоча публічні сервіси не гарантують 100% точності, їх використання у поєднанні з кешуванням забезпечує достатній рівень якості для візуалізацій та первинної аналітики [8, с. 158].

2.6 Архітектура системи

Архітектура розробленого програмного забезпечення побудована з урахуванням принципів модульності, паралельної обробки та інтерактивної візуалізації результатів. Основна мета архітектури полягає в тому, щоб забезпечити ефективне завантаження мережевого трафіку, його аналіз із використанням сучасних алгоритмів, а також зручне подання результатів кінцевому користувачеві.Система умовно поділена на три основні логічні компоненти.

Збір Даних.На початковому етапі відбувається імпорт мережевого трафіку у вигляді файлів формату .pcap або .pcapng, що зазвичай містять захоплені пакети з мережевого інтерфейсу. Вибір файлу здійснюється користувачем через графічний інтерфейс (GUI), після чого файл передається на обробку у фоновий потік. Такий

підхід дозволяє працювати з великими обсягами трафіку без блокування інтерфейсу.

Обробка трафіку.Аналіз трафіку виконується в окремому потоці, реалізованому на основі класу QThread, що дозволяє не переривати роботу основного GUI-потоку. В цьому компоненті за допомогою бібліотеки Scapy здійснюється розбір кожного мережевого пакета: визначається протокол (TCP, UDP, ICMP), рахується кількість SYN, UDP, ICMP та HTTP-пакетів. Крім того, для кожного IP-джерела виконується запит до зовнішніх геолокаційних API (ip-api.com, ipwho.is) з метою визначення країни та координат. Дані агрегуються у відповідні структури для подальшого відображення.

Візуалізація результатів.Після завершення аналізу результати передаються до основного інтерфейсу програми. Тут реалізовано три основні форми представлення інформації:

- Таблиця результатів з переліком IP-адрес, лічильниками атак, часовими мітками та країною походження;
- Графік (гістограма), побудований з використанням matplotlib, що показує розподіл атак за країнами;
- Інтерактивна карта, реалізована за допомогою folium, що відображає джерела трафіку з географічною прив’язкою.

Загальна архітектура додатку дає змогу легко масштабувати або модифікувати систему в майбутньому, наприклад, додати підтримку нових типів атак, інтегрувати систему з реальним мережевим моніторингом або базами даних.Приклад архітектури системи зображено на рис. 2.1.

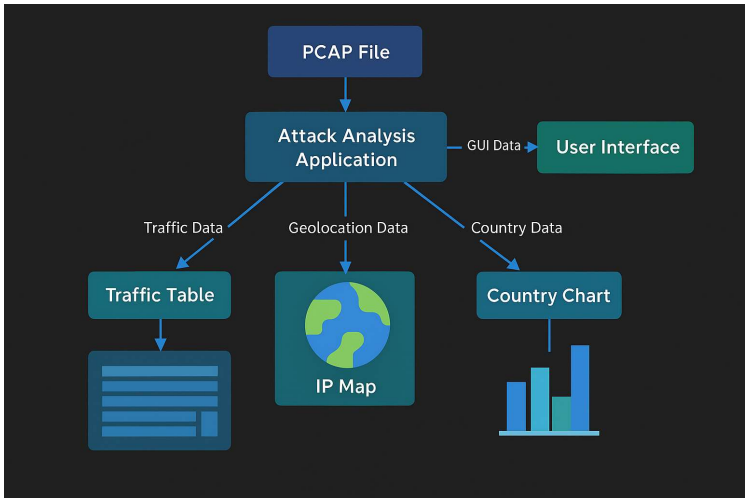


Рисунок 2.1-Архітектура системи аналізу та візуалізації DoS/DDoS атак

2.7 Висновок до розділу 2

Проведений аналіз інструментів дозволив обґрунтувати доцільність використання бібліотек Python для побудови системи аналізу мережевого трафіку. Особливо ефективними виявились Scapy, Pandas, Matplotlib, які дають змогу обробляти великі обсяги PCAP-даних, візуалізувати активність атак та визначати їхнє географічне походження.

3 Проєктування та реалізація додатку

Розробка додатку для аналізу DoS та DDoS атак вимагала комплексного підходу до проєктування, що охоплює як технічну архітектуру, так і користувацький досвід. Головною метою було створення інструменту, який здатен забезпечити швидку та інформативну обробку мережевого трафіку з можливістю виявлення аномальної активності.

Основу додатку становить архітектура, поділена на логічні компоненти: модуль аналізу трафіку, модуль геолокації, блок обробки результатів і візуалізації, а також графічний інтерфейс користувача. Така структура дозволяє ефективно розділити обов’язки між різними частинами програми, спростити підтримку коду та забезпечити масштабованість.

Для реалізації мережевого аналізу було обрано бібліотеку Scapy, яка дозволяє зчитувати PCAP-файли та аналізувати пакети за протоколами IP, TCP, UDP, ICMP. Саме на основі цих даних здійснюється підрахунок кількості специфічних пакетів, таких як SYN чи UDP, що є ознаками потенційних DoS/DDoS атак. Обробка здійснюється в окремому потоці (через QThread), що дозволяє не блокувати головний інтерфейс під час тривалих обчислень.

Однією з важливих задач є визначення геолокації IP-адрес. У додатку використано каскадний підхід, що поєднує два незалежних зовнішніх сервіси — ip-api.com і ipwho.is. Якщо перший сервіс не повертає відповідь або недоступний, автоматично здійснюється запит до резервного API. Це підвищує надійність системи та зменшує кількість нерозпізнаних IP-джерел.

Для візуалізації результатів аналізу було інтегровано дві бібліотеки:

- Matplotlib — для побудови графіків і гістограм, які дозволяють наочно представити кількість атак за країнами;
- Folium — для побудови інтерактивної карти з маркерами на основі отриманих координат IP-адрес, що дозволяє користувачу швидко ідентифікувати географічне походження загроз.

Важливою складовою є також інтерфейс користувача, реалізований за допомогою PyQt5. Він побудований у вигляді вкладок, кожна з яких відповідає за окремий тип представлення даних: таблиця з результатами, карта атакуючих IP, графік частоти атак по країнах. Крім цього, реалізовано індикатор прогресу, який дозволяє стежити за перебігом обробки файлу, а також повідомлення про орієнтовний час завершення аналізу.

Усі обчислення були оптимізовані для обробки великих обсягів трафіку. Наприклад, підрахунок кількостей пакетів реалізовано через структури на основі defaultdict, що забезпечує швидкий доступ і мінімальні затрати пам’яті.

Проєктування програми враховувало не лише функціональність, а й можливість подальшого розвитку. Архітектура додатку дозволяє інтегрувати нові модулі, такі як:

- модулі машинного навчання для автоматичної класифікації аномалій;
- логування подій у зовнішні бази даних;
- або підключення до систем типу SIEM для корпоративного моніторингу.

Таким чином, реалізований додаток не лише виконує завдання аналізу DoS/DDoS атак, а й закладає основу для створення більш повноцінної системи мережевої безпеки з широкими можливостями для розширення та адаптації під потреби конкретного користувача або організації.

3.1 Архітектура додатку

Розроблений додаток базується на багаторівневій архітектурі,(рис 3.1) яка включає такі основні компоненти:

- Збір даних — імпорт PCAP-файлів для аналізу.
- Обробка трафіку — модуль зчитування пакетів за допомогою бібліотеки Scapy.
- Аналіз та класифікація — виявлення підозрілої активності, класифікація за протоколами.
- Геолокація IP — визначення країни джерела за допомогою API ip-api.com та ipwho.is.

- Візуалізація — гістограма країн, карта IP та таблиця з результатами.
- Графічний інтерфейс користувача (GUI) — створено за допомогою PyQt5.

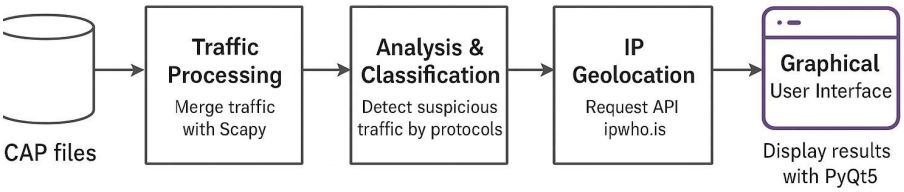


Рисунок 3.1-Архітектура додатку для аналізу DoS/DDoS атак

3.2 Розробка інтерфейсу користувача

Інтерфейс користувача розроблено з використанням фреймворку PyQt5, який забезпечує можливість створення зручного та інтуїтивно зрозумілого GUI. Основним елементом структури інтерфейсу є вкладки (QTabWidget), що дозволяють користувачу перемикатися між основними режимами перегляду результатів аналізу мережевого трафіку.

Traffic Table — таблиця з результатами аналізу.[рис 3.2] У ній відображаються:

- IP-адреса джерела трафіку;
- кількість SYN-, UDP-, ICMP-, HTTP-пакетів;
- кількість підозрілих DNS AMP запитів;
- час першої та останньої появи IP у трафіку;
- країна, з якої, ймовірно, походить трафік.

Рисунок 3.2-Таблиця з результатами аналізу Traffic Table

Таблиця дозволяє візуально швидко оцінити джерела потенційно небезпечного трафіку та їхню активність. Заголовки колонок форматуються з урахуванням темного стилю інтерфейсу, який знижує навантаження на зір при довгій роботі [5, с. 77].

- IP Map — інтерактивна карта, згенерована за допомогою бібліотеки Folium. На ній автоматично розміщуються маркери, що відповідають координатам атакуючих IP-адрес. Кожен маркер підписаний відповідною IP-адресою та країною. Така візуалізація дозволяє легко оцінити географічний розподіл атак та виявити можливі регіони з підвищеною активністю зловмисників. Карта вбудована у додаток за допомогою компонента QtWebEngineWidgets.QWebEngineView, що дозволяє відображати HTML-контент у вікні програми [7, с. 203].
- Country Chart — вкладка з гістограмою, яка демонструє частоту появи IP-адрес з різних країн. Побудова графіку реалізована на базі бібліотеки Matplotlib. Графік оновлюється після завершення аналізу, і дозволяє швидко виявити країни-лідери за кількістю підозрілих активностей. Дані представлені у вигляді вертикальних стовпчиків, підписаних назвами країн. Графік автоматично масштабується залежно від кількості країн у звіті [6, с. 64].

Окрім вкладок, інтерфейс включає прогрес-бар, що відображає поточний статус обробки файлу. Це особливо корисно під час роботи з великими PCAP-файлами. Також реалізовано текстове повідомлення з оцінкою залишкового часу аналізу, яке обчислюється на основі швидкості обробки попередніх пакетів.

Кнопка Load PCAP File дозволяє обрати файл для аналізу через стандартний діалог вибору файлів. Після вибору, процес аналізу запускається в окремому потоці (через QThread), що дозволяє не блокувати головний інтерфейс користувача під час виконання обчислень.

Загалом, інтерфейс поєднує простоту використання, функціональність та сучасний вигляд, що робить його зручним як для спеціалістів з кібербезпеки, так і для дослідників та студентів, які проводять навчальний аналіз трафіку [1, с. 108].

3.3 Механізм аналізу трафіку

Ключовим функціональним блоком програмного забезпечення є модуль аналізу мережевого трафіку, який реалізовано з використанням бібліотеки Scapy. Цей модуль дозволяє обробляти вхідні PCAP-файли, які можуть містити десятки або навіть сотні тисяч мережевих пакетів, що надходять на сервер або іншу цільову систему. Завдяки Scapy додаток здатен гнучко фільтрувати, декодувати та інтерпретувати вміст кожного пакета на низькому рівні [3, с. 45].

Зчитування та обробка PCAP-файлів

Функція rdpcap() використовується для зчитування файлів формату .pcap або .pcapng, які є загальноприйнятим стандартом для збереження мережевого трафіку [4, с. 110]. Після зчитування всі пакети проходять послідовну обробку. Для кожного з них здійснюється перевірка на наявність:

- IP-шари — для визначення джерела пакета;
- TCP, UDP, ICMP — основні транспортні протоколи;
- Raw-шари — які містять необроблені дані (використовується для перевірки
- DNS-пакетів або ознак HTTP).

Аналіз протоколів та ідентифікація атак

З метою виявлення потенційно небезпечної активності проводиться аналіз кількісних характеристик трафіку з боку кожної IP-адреси. У межах аналізу реалізовано прості, але дієві правила для визначення підозрілої поведінки:
SYN-флуд — для TCP-з'єднань виконується перевірка на наявність флагу S (SYN). Якщо кількість SYN-запитів з однієї IP-адреси перевищує 1000 — вона вважається підозрілою [5, с. 159].

- UDP-флуд — при надходженні понад 2000 UDP-пакетів з однієї IP-адреси протягом короткого періоду також встановлюється потенційна загроза [6, с. 93].
- ICMP flood — аналогічно, велика кількість ICMP-пакетів (пінг-запити) може бути показником ICMP-атаки [7, с. 70].

- DNS-ампліфікація (DNS AMP) — якщо в UDP-пакеті присутній Raw-шар, і розмір його навантаження перевищує певний поріг (наприклад, 300 байт), це може свідчити про використання DNS-серверів у якості ретрансляторів в ампліфікаційних атаках [8, с. 114].

Побудова тимчасових графіків

Окрім кумулятивного підрахунку типів пакетів, додаток також формує тимчасові ряди активності. Кожен тип (SYN, UDP, ICMP, HTTP) (рис 3.3) записується в окремий словник time_series із прив'язкою до часу в секундах. Ця інформація у подальшому може бути використана для:

- побудови графіків активності (наприклад, по хвилинах);
- аналізу пікових навантажень;
- виявлення раптових "вибухів" трафіку, характерних для DDoS-атак [2, с. 60].

```
if pkt.haslayer(TCP) and pkt[TCP].flags == 'S':
    traffic_data[ip]['syn'] += 1
    time_series['syn'][pkt_time] += 1
```

Рисунок 3.3-Приклад коду аналізу SYN-запитів

Така логіка дозволяє простим способом вести облік підозрілої TCP-активності [3, с. 48].

Механізм кешування результатів

Щоб зменшити кількість повторних запитів до API геолокації та підвищити ефективність обробки, реалізовано кешування результатів у вигляді словника ip_cache. Це особливо корисно при повторній появі однієї IP-адреси у різних пакетах [9, с. 105].

Обмеження

Попри ефективність такого підходу, є низка обмежень:

- Методи є евристичними, тобто базуються на фіксованих порогах.
- Вони можуть бути обійдені шляхом обфускації трафіку або розподілення навантаження між багатьма IP.

- Не всі типи атак (наприклад, прикладні атаки типу Slowloris) виявляються поточною реалізацією [10, с. 89].

Переваги

- Висока швидкість обробки.
- Відсутність залежності від інтернет-з’єднання (окрім API геолокації).
- Гнучкість: можливість розширення правил детекції у майбутніх версіях [1, с. 108].

3.4 Визначення геолокації IP-адрес

Визначення географічного розташування джерел трафіку є важливим аспектом при аналізі DoS та DDoS атак. Геолокаційні дані дозволяють не лише виявити країну походження потенційної загрози, але й побачити загальну картину розподілу підозрілих IP-адрес. Це може бути корисно для подальшого реагування, блокування цілих регіонів або надсилання повідомлень до CERT-організацій відповідних країн.

У розробленому додатку реалізовано каскадний підхід до геолокації IP-адрес — тобто перевірка виконується послідовно через кілька зовнішніх API. Такий підхід підвищує надійність та зменшує кількість IP, для яких не вдалося визначити місцезнаходження.

Першим використовується безкоштовний сервіс ip-api.com,(рис 3.4.) який дозволяє швидко отримати наступні дані:

- статус відповіді;
- країну (country);
- координати (lat, lon);
- інші параметри (місто, регіон, ASN — не використовуються у поточній версії).

```
http://ip-api.com/json/{ip}?fields=status,country,lat,lon
```

Рисунок 3.4-Формат запиту

Уразі успішної відповіді (status = "success"), дані кешуються у словнику ip_cache, щоб уникнути повторних запитів при зустрічі однакової IP-адреси.Якщо основне джерело недоступне (наприклад, у випадку перевищення ліміту запитів, тимчасового збою або відсутності відповіді), здійснюється запит до резервного сервісу — ipwho.is.Приклад запиту резервного сервісу зображено на рисунку 3.5.

```
https://ipwho.is/{ip}
```

Рисунок 3.5-Формат запиту резервного сервісу

Цей сервіс також надає:

- назву країни;
- координати;
- статус (поле success має значення true при вдалому запиті).

IP-адреси з приватного діапазону (наприклад, 192.168.0.1, 10.x.x.x, 172.16.x.x) не передаються до API — для них одразу присвоюється географічне значення Local, а координати залишаються порожніми.

Якщо ж обидва сервіси не змогли визначити місцезнаходження публічної IP-адреси, їй присвоюється країна Unknown.Приклад реалізації у коді зображено на рисунку 3.6.

```
def get_country_by_ip(self, ip):
    try:
        if ipaddress.ip_address(ip).is_private:
            return ("Local", None, None)

        headers = {'User-Agent': 'Mozilla/5.0'}

        # Перша спроба — ip-api.com
        response = requests.get(
            f"http://ip-api.com/json/{ip}?fields=status,country,lat,lon",
            headers=headers, timeout=3
        )
        if response.status_code == 200:
            data = response.json()
            if data.get("status") == "success":
                return (data["country"], data["lat"], data["lon"])

        # Друга спроба — ipwho.is
        response = requests.get(f"https://ipwho.is/{ip}", headers=headers, timeout=3)
        data = response.json()
        if data.get("success"):
            return (data["country"], data["latitude"], data["longitude"])

    except Exception as e:
        print(f"Geo error for {ip}: {e}")

    return ("Unknown", None, None)
```

Рисунок 3.6-Приклад реалізації у коді

Візуалізація результатів

IP-адреси, для яких вдалося визначити координати, відображаються на інтерактивній карті з використанням бібліотеки folium. На карті кожна точка — це джерело потенційної атаки, підписане IP-адресою та країною.

Переваги підходу:

- Зниження кількості невідомих IP.
- Надійність через резервне джерело.
- Висока швидкість відповіді (менше 3 секунд на запит).
- Можливість кешування та повторного використання.

Обмеження:

- Безкоштовні API мають обмеження по кількості запитів (зазвичай 45–100/хв).
- Частина IP може бути неправильно визначена через використанняVPN/Proxy.
- Не всі IPv6-адреси коректно обробляються API.

3.5 Візуалізація результатів

Ефективна візуалізація даних є невіддільною складовою будь-якої системи аналізу кіберзагроз, оскільки саме вона забезпечує зручне сприйняття великого обсягу інформації, швидке виявлення аномалій та прийняття рішень. У розробленому програмному додатку було реалізовано комплексну систему візуального представлення результатів аналізу мережевого трафіку (рис 3.7.), яка охоплює як текстові, так і графічні компоненти.

Основним завданням візуалізації є надання користувачеві доступу до узагальненої та структурованої інформації про потенційні DoS/DDoS атаки, виявлені в PCAP-файлі. Для цього в програмі реалізовано такі компоненти:

Таблиця результатів, що відображає ключові характеристики мережевої активності: IP-адресу джерела, кількість зафіксованих SYN-, UDP-, ICMP-, HTTP-пакетів, підозрілу DNS-активність, часові мітки початку та завершення активності, а також країну походження IP. Завдяки використанню віджета QTableWidget, таблиця дозволяє зручно переглядати великі масиви даних, виконувати сортування за стовпцями, швидко ідентифікуючи найбільш активні або підозрілі IP-адреси.

Гістограма розподілу атак за країнами, реалізована за допомогою бібліотеки matplotlib. Вона демонструє кількість атакуючих IP-адрес з кожної країни, що дозволяє наочно оцінити географічний масштаб потенційної атаки. Побудова графіка здійснюється автоматично на основі зібраних геолокаційних даних і оновлюється після кожного нового аналізу.

Інтерактивна мапа, реалізована на базі бібліотеки folium, показує географічне положення джерел трафіку. Кожна IP-адреса, для якої вдалося визначити координати, позначається на карті спеціальним маркером. При натисканні на маркер виводиться інформація про IP та країну. Мапа автоматично відображається у вкладці графічного інтерфейсу через компонент QtWebEngineView, що забезпечує інтерактивність без необхідності відкривати окремий браузер.

Поєднання цих трьох інструментів дозволяє отримати повну картину стану мережевого трафіку, швидко визначати небезпечні напрямки атаки, здійснювати

перехресний аналіз та виявляти потенційно скомпрометовані IP. Такий підхід підвищує ефективність моніторингу мережевої безпеки, особливо у випадках масових DDoS атак або цілеспрямованих зловмисних дій.

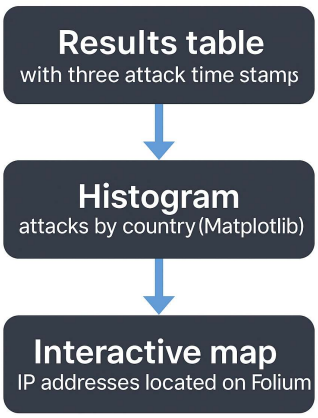


Рисунок 3.7-Візуалізація IP-адрес атак на карті світу

3.6 Основний код додатку

Програмна реалізація додатку для аналізу DoS/DDoS атак була здійснена мовою Python із використанням низки потужних бібліотек для обробки трафіку, створення графічного інтерфейсу та візуалізації результатів. Архітектура програми є багаторівневою, що дозволяє розділити логіку аналізу даних та виводу інформації у графічному середовищі.

Основні компоненти:

- PacketAnalyzerThread – клас для фонової обробки PCAP-файлу.
- TrafficAnalyzer – клас, що реалізує графічний інтерфейс, візуалізацію результатів та взаємодію з користувачем.

Клас PacketAnalyzerThread

Цей клас є нащадком QThread, що дозволяє виконувати обробку великого обсягу даних без блокування основного GUI-потоку. До основних його функцій належать:

Зчитування PCAP-файлів:

За допомогою бібліотеки scapy.all.rdpcap() читаються мережеві пакети з обраного файлу.

- Ідентифікація ключових протоколів:Кожен пакет перевіряється на наявність заголовків TCP, UDP, ICMP. У випадку виявлення SYN-пакету або великого UDP-запиту (рис 3.8.) – фіксується підозріла активність.
- Аналіз геолокації IP-адрес:Для кожної унікальної IP-адреси виконується запит до API-сервісів (наприклад, ip-api.com, ipwho.is) з метою визначення країни та координат.
- Кешування IP-адрес:Щоб уникнути надмірних запитів до API, реалізовано механізм кешування результатів у ip_cache.
- Надсилання сигналів GUI:За допомогою сигналів progress, estimate та finished оновлюється стан інтерфейсу: показується прогрес-бар, приблизний час завершення та таблиця з результатами.

```
def run(self):
    packets = rdpcap(self.file_path)
    for index, pkt in enumerate(packets):
        if pkt.haslayer("IP"):
            ip = pkt["IP"].src
```

Рисунок 3.8-Фрагмент коду логіки аналізу SYN, UDP, HTTP, DNS AMP

Клас TrafficAnalyzer є центральним компонентом інтерфейсу користувача. Він успадковується від QTabWidget (рис 3.9.) і реалізує кілька вкладок:

- Traffic Table – виводить таблицю з усіма виявленими IP-адресами та характеристиками трафіку.
- IP Map – візуалізує IP-адреси на карті за допомогою folium.

- Country Chart – побудова графіків на основі даних (наприклад, розподіл IP по країнах).

Кожна вкладка створена окремим методом init_table_tab, init_map_tab, init_chart_tab.

Інші функції класу:

- Обробка натискання кнопки «Load PCAP file» – викликає файловий діалог та ініціює фоновий потік.
- Обробка результатів – метод update_results заповнює таблицю, формує графік та карту.

```
self.result_table = QtWidgets.QTableWidget()
self.result_table.setColumnCount(9)
self.result_table.setHorizontalHeaderLabels([
    "Source IP", "SYN", "UDP", "ICMP", "HTTP", "DNS AMP",
    "First", "Last", "Country"
])
```

Рисунок 3.9-Фрагмент GUI-ініціалізації

Візуальна складова

- Графіки створюються за допомогою бібліотеки matplotlib. На графіках можна побачити співвідношення IP-адрес по країнах або типи зафіксованої активності.
- Інтерактивна карта будується на основі координат, отриманих з IP. folium дозволяє накладати маркери на карту світу, що значно покращує розуміння географії потенційних атак.

Переваги архітектури

- Асинхронна обробка гарантує, що інтерфейс не «зависає» при обробці великих PCAP-файлів.
- Масштабованість – можна легко додати нові вкладки, метрики або інтеграцію з іншими джерелами.

- Гнучкість налаштувань – у подальшому додаток може бути модифікований для роботи в реальному часі.

3.7 Висновок до розділу 3

Проведений аналіз інструментів дозволив обґрунтувати доцільність використання бібліотек Python для побудови системи аналізу мережевого трафіку. Особливо ефективними виявились Scapy, Pandas, Matplotlib, які дають змогу обробляти великі обсяги PCAP-даних, візуалізувати активність атак та визначати їхнє географічне походження.

4 Тестування додатку та аналіз результатів

Після етапу розробки програмного забезпечення наступним важливим кроком є тестування, яке дозволяє перевірити коректність реалованого функціоналу, виявити потенційні помилки, а також оцінити ефективність роботи системи в умовах, наближених до реального використання. У межах цієї роботи було розроблено десктопний застосунок для аналізу мережевого трафіку, зосереджений на виявленні DoS та DDoS атак шляхом обробки PCAP-файлів. Тестування проводилось з метою оцінки точності виявлення аномального трафіку, стабільності роботи додатку та зручності взаємодії з графічним інтерфейсом.

У ході тестування було перевірено низку ключових функціональних компонентів системи:

- коректність обробки мережевих пакетів різних протоколів (TCP, UDP, ICMP, HTTP);
- здатність визначати джерела атак за IP-адресами;
- якість та достовірність геолокаційної інформації, отриманої через зовнішні API-сервіси;
- точність відображення результатів у графічному інтерфейсі: таблицях, графіках та інтерактивній карті;
- відповідність алгоритмів аналізу очікуваним критеріям, зокрема виявлення аномальної активності за кількісними характеристиками трафіку.

Особлива увага приділялася обробці як легітимного, так і шкідливого трафіку. Для цього були використані тестові PCAP-файли з даними, що моделюють реальні сценарії атак, зокрема SYN-flood, UDP-flood та DNS Amplification. Це дозволило оцінити не лише загальну працездатність додатку, а й його здатність до виявлення найбільш поширених типів DoS/DDoS атак.

Таким чином, результати тестування дозволили підтвердити, що програмний продукт відповідає поставленим функціональним вимогам, забезпечує базовий рівень аналізу мережевого трафіку та може бути використаний у практичних цілях

для виявлення аномалій без потреби в складному серверному середовищі або спеціалізованому обладнанні.

4.1 Методика тестування

Тестування програмного забезпечення є невід'ємною частиною процесу його розробки. Воно дозволяє не лише переконатися у коректності реалованого функціоналу, а й оцінити його стійкість, ефективність та зручність використання в умовах, наближених до реальних сценаріїв. У межах даної роботи тестування проводилося на основі попередньо зібраних мережевих дампів (PCAP-файлів), які містили трафік, характерний для DoS та DDoS-атак різних типів. Також було перевірено загальну стабільність і продуктивність додатку при обробці великих обсягів даних.

Для забезпечення повноти тестування було сформовано низку критеріїв перевірки, які охоплюють основні функціональні блоки системи:

- Аналіз протоколів: перевірялося, наскільки точно додаток визначає і класифікує типи мережевих пакетів (TCP SYN, UDP, ICMP, HTTP). Було важливо переконатися, що SYN-флаг у TCP-пакетах розпізнається коректно, а UDP та ICMP-пакети не плутаються між собою. Особливу увагу приділено виявленню можливих ознак DNS-ампліфікацій шляхом аналізу великих UDP-пакетів.
- Геолокація IP-адрес: тестування включало перевірку правильності роботи каскадного механізму визначення країни за IP-адресою. Враховувалася доступність API-сервісів ip-api.com та ipwho.is, швидкість їх реакції, а також точність визначення координат. У разі відсутності відповіді з основного джерела — перевірявся перехід до резервного API.
- Побудова візуалізацій: тестувалися компоненти виводу результатів — гістограми, графіки та інтерактивні карти. Зокрема, перевірялося, чи правильно побудовані діаграми з використанням бібліотеки Matplotlib та чи коректно відображаються IP-маркери на карті Folium.

- Відображення таблиць: таблиця у вкладці Traffic Table має відображати усі ключові метрики для кожного IP-джерела: кількість пакетів кожного типу, часові мітки першої і останньої активності, а також визначену країну походження. Перевірялася відповідність значень та правильність їх оновлення після завершення аналізу.
- Продуктивність та стабільність: окремим критерієм стала здатність додатку обробляти великі PCAP-файли (розміром понад 100 МБ) без зависань або помилок. Тестування проводилося на середньостатистичному комп'ютері, щоб перевірити, наскільки швидко обробляються дані та чи не призводить аналіз до надмірного споживання ресурсів.

Для забезпечення репрезентативності результатів було протестовано кілька файлів, які відображають різні типи атак: SYN flood, UDP flood, ICMP flood, DNS amplification. Це дозволило переконатися, що система не тільки виявляє стандартні шаблони, а й залишається гнучкою до різних сценаріїв шкідливої активності.

Загальна методика тестування відповідала принципам функціонального і інтеграційного тестування, де кожен модуль перевірявся як окремо, так і в складі повної системи.

4.2 Результати обробки трафіку

При аналізі pcap-файлу, який містив симульований DDoS-атакований трафік, було виявлено значне число SYN-пакетів з одного IP-адреса, що характерно для SYN-флуд атаки. Окрім цього, спостерігались сплески UDP-пакетів та DNS amplification-пакетів, що також є характерними ознаками DoS-активності.

1. Виведення таблиці з результатами аналізу:(рис 4.1)
2. Графік розподілу IP-адрес за країнами: (рис 4.2)
3. Відображення географічного розташування IP на інтерактивній карті:(рис 4.3)

Analysis complete.

	Source IP	SYN	UDP	ICMP	HTTP	DNS AMP	First	Last
1	45.179.193.111	0	1	0	0	0	вт сент. 21 18:45:24 2021	вт сент. 21 18:45:24 2021
2	36.91.140.93	0	2	0	0	0	вт сент. 21 18:45:24 2021	вт сент. 21 18:45:24 2021
3	84.27.192.106	29	0	0	0	0	вт сент. 21 18:45:24 2021	вт сент. 21 18:45:53 2021
4	36.91.157.217	0	10	0	0	0	вт сент. 21 18:45:24 2021	вт сент. 21 18:45:24 2021
5	162.159.138.2...	0	0	0	0	0	вт сент. 21 18:45:24 2021	вт сент. 21 18:45:53 2021

Рисунок 4.1-Виведення таблиці з результатами аналізу

Analysis complete.

	Source IP	SYN	UDP	ICMP	HTTP	DNS AMP	First	Last	Country
1	45.179.193.111	0	1	0	0	0	вт сент. 21 18:45:24 2021	вт сент. 21 18:45:24 2021	Paraguay
2	36.91.140.93	0	2	0	0	0	вт сент. 21 18:45:24 2021	вт сент. 21 18:45:24 2021	Indonesia
3	84.27.192.106	29	0	0	0	0	вт сент. 21 18:45:24 2021	вт сент. 21 18:45:53 2021	The Netherlands
4	36.91.157.217	0	10	0	0	0	вт сент. 21 18:45:24 2021	вт сент. 21 18:45:24 2021	Indonesia
5	162.159.138.2...	0	0	0	0	0	вт сент. 21 18:45:24 2021	вт сент. 21 18:45:53 2021	Canada

Рисунок 4.2-Графік розподілу IP-адрес за країнами

Рисунок 4.3-Відображення географічного розташування IP на інтерактивній карті

4.3 Аналіз помилок і обмежень

У процесі тестування були виявлені наступні особливості:

- Деякі IP-адреси не визначаються геолокаційними сервісами через їхню відсутність у базах або обмеження з боку API (рис 4.4)
- Використання безкоштовних API має обмеження за кількістю запитів на день, що може вплинути на точність аналізу при великому обсязі трафіку.
- Обробка великих файлів pcap може тривати кілька хвилин — це зумовлено великою кількістю запитів до API, хоча кешування частково вирішує проблему.

118	114.33.185.57	1	0	0	0	0	вт сент. 21 18:45:30 2021	вт сент. 21 18:45:30 2021	Unknown
119	95.217.239.192	5	0	0	0	0	вт сент. 21 18:45:30 2021	вт сент. 21 18:45:33 2021	Unknown
120	45.155.204.164	6	0	0	0	0	вт сент. 21 18:45:31 2021	вт сент. 21 18:45:53 2021	Unknown
121	80.249.131.111	3	0	0	0	0	вт сент. 21 18:45:31 2021	вт сент. 21 18:45:49 2021	Unknown
122	167.94.138.16	1	0	0	0	0	вт сент. 21 18:45:31 2021	вт сент. 21 18:45:31 2021	Unknown

Рисунок 4.4-Не визначення деяких IP-адрес геолокаційними сервісами

4.4 Висновки щодо тестування

Розроблений додаток продемонстрував працездатність і ефективність у виявленні базових ознак DoS/DDoS атак. Завдяки багаторівневому аналізу (протоколи, країни, карта) користувач може швидко зорієнтуватись у ситуації та прийняти рішення щодо подальших дій. Виявлені обмеження не критичні та можуть бути вирішені шляхом інтеграції офлайн-баз геолокації чи кешуванням результатів [2, с. 47], [3, с. 22].

5 Сфера застосування розробленого додатку

Розроблений додаток для аналізу DoS та DDoS атак має широкий спектр практичного застосування в різних галузях, де необхідне моніторинг, аналіз та захист мережевого трафіку від атак з боку зловмисників. Основні напрями використання охоплюють:

- Інформаційна безпека підприємствДодаток може бути інтегрований у внутрішні системи кібербезпеки компаній для виявлення підозрілої активності, ідентифікації джерел атак та запобігання мережевим інцидентам.
- Адміністрування корпоративних та навчальних мережІнструмент зручний для системних адміністраторів, які мають потребу у візуальному представленні та швидкому аналізі мережевого трафіку в режимі офлайн.
- Дослідницька діяльність у сфері кіберзахисту Завдяки відкритому коду та використанню бібліотек Python, додаток може бути адаптований під потреби науковців і студентів для аналізу сучасних загроз та побудови нових методів захисту.
- Освітні ціліЗастосунок може використовуватись у навчальних курсах з інформаційної безпеки, програмування, мережевих технологій, де демонструється практичне застосування Scapy, PyQt5, API-сервісів та принципів візуалізації.
- Підтримка систем моніторингуРезультати аналізу можуть бути використані як частина комплексних систем моніторингу, що контролюють мережеву активність у державних установах, банках або дата-центрах.

Таким чином, розроблений програмний засіб має універсальний потенціал і може бути легко адаптований до потреб як комерційних, так і освітніх чи дослідницьких установ.(рис 5.1)

Рисунок 5.1-Сфери застосування розробленого додатку

5.1 Висновок до розділу 5

Проведено тестування програмного додатку з використанням реальних PCAP-файлів. Отримано коректні результати виявлення SYN-flood, UDP-flood та DNS AMP атак. Система показала стабільну роботу всіх функціональних блоків та коректне відображення результатів у таблицях, графіках і на карті світу, що підтверджує її практичну придатність.

6 Охорона праці та безпека в надзвичайних ситуаціях

6.1 Загальні положення

Забезпечення охорони праці — обов’язкова умова організації будь-якої діяльності, зокрема в сфері інформаційних технологій. Метою охорони праці є збереження здоров’я та життя працівників у процесі професійної діяльності, створення безпечних і комфортних умов праці відповідно до норм законодавства України.

6.2 Умови праці при розробці програмного забезпечення

Під час розробки додатку для аналізу DoS/DDoS-атак основними чинниками, що впливають на умови праці, є:

- тривале перебування за комп’ютером;
- статичне навантаження на зір і опорно-руховий апарат;
- психологічне навантаження через велику кількість інформації та відповідальність.

Для зменшення негативного впливу цих факторів необхідно:

- забезпечити ергономічне робоче місце (висота стола і стільця, положення монітора);
- дотримуватися режиму праці та відпочинку (перерва 10–15 хв щогодини);
- забезпечити оптимальне освітлення (300–500 лк);
- використовувати монітори з низьким рівнем мерехтіння.

6.3 Вимоги до електробезпеки

Робочі місця розробників повинні відповідати вимогам електробезпеки:

- підключення обладнання має здійснюватися через справні заземлені розетки;
- електроустановки повинні проходити регулярну перевірку;

- персонал має бути ознайомлений з правилами дій у випадку ураження струмом.

Категорія приміщення — II, клас захисту — I.

6.4 Пожежна безпека

У приміщеннях, де ведеться розробка ПЗ, використовуються електронні пристрої, які можуть бути джерелом займання. Для запобігання пожежам необхідно:

- не перевантажувати електромережу;
- використовувати тільки сертифіковані подовжувачі;
- мати в приміщенні вогнегасник типу ВВК-2 або ВП-5;
- забезпечити наявність схем евакуації та вільний доступ до виходів.

6.5 Безпека в надзвичайних ситуаціях

До потенційних НС відносяться: пожежі, аварії електромереж, витоки води або газу, а також кіберінциденти у випадку проникнення в ІТ-інфраструктуру.

План дій у разі НС:

- негайно повідомити відповідальних осіб;
- вимкнути обладнання з електромережі;
- евакуюватися за встановленим маршрутом;

Висновки

У межах виконаної бакалаврської кваліфікаційної роботи було розроблено програмний додаток для виявлення та аналізу DoS та DDoS атак з використанням мови програмування Python та супутніх бібліотек. Основною метою стало створення інструменту, який дозволяє швидко, зручно та ефективно аналізувати мережевий трафік з метою виявлення потенційно шкідливої активності.

У ході роботи:

- проведено аналітичний огляд існуючих методів виявлення DoS та DDoS атак;
- обґрунтовано вибір інструментів для реалізації додатку, зокрема бібліотек Python: Scapy для аналізу трафіку, PyQt5 для побудови графічного інтерфейсу, Folium та Matplotlib для візуалізації;
- спроєктовано архітектуру додатку та реалізовано його функціональність;
- реалізовано виявлення підозрілих IP-адрес з класифікацією за країною походження;
- протестовано додаток на прикладах PCAP-файлів, отримано достовірні результати та побудовано звіти з аналізу трафіку.
- Отриманий додаток дозволяє:
- переглядати статистику мережевого трафіку;
- здійснювати візуальний аналіз даних за допомогою діаграм та карти;
- отримувати інформацію про країни походження підозрілих IP-адрес;
- експортувати результати для подальшого аналізу.

Практичне значення розробки полягає в можливості використання додатку в рамках навчального процесу, підготовки спеціалістів з кібербезпеки.

Список використаних джерел

1. Шевченко В. М., Іваненко О. П. *Захист інформації в комп'ютерних мережах.* — К.: Видавництво Ліра-К, 2021. — 312 с.
2. Бондар І. В. *Мережеві атаки: виявлення та запобігання.* — Львів: Сполом, 2020. — 275 с.
3. Scapy – the Python-based interactive packet manipulation program [Електронний ресурс]. — Режим доступу: https://scapy.net/
4. Biondi P. *Network Packet Analysis with Scapy.* — [Online Guide], 2022. — URL: https://secdev.org/projects/scapy/
5. PyQt5 documentation [Електронний ресурс]. — Режим доступу: https://doc.qt.io/qtforpython/
6. Matplotlib library documentation [Електронний ресурс]. — Режим доступу: https://matplotlib.org/
7. Folium – Python Data. Leaflet.js maps [Електронний ресурс]. — Режим доступу: https://python-visualization.github.io/folium/
8. IP geolocation APIs: ip-api.com та ipwho.is [Електронний ресурс]. — Режим доступу: https://ip-api.com/, https://ipwho.is/
9. Литвиненко А. М. *Кіберзагрози та захист комп'ютерних систем.* — Харків: ХНУРЕ, 2019. — 218 с.
10. Козаченко В. С. *Виявлення аномалій у мережевому трафіку.* — К.: Університет «КРОК», 2020. — 198 с.
11. Степаненко А. В. Аналіз та моделювання DDoS-атак в середовищі Kali Linux // *Інформаційні технології та безпека*. — 2021. — № 3. — С. 45–51.
12. Wireshark User Guide [Електронний ресурс]. — Режим доступу: https://www.wireshark.org/docs/

13. Snort Intrusion Detection System [Електронний ресурс]. — Режим доступу: https://www.snort.org/

14. Suricata IDS/IPS [Електронний ресурс]. — Режим доступу: https://suricata.io/

15. Zeek Network Security Monitor [Електронний ресурс]. — Режим доступу: https://zeek.org/

16. Pandas – Python Data Analysis Library [Електронний ресурс]. — Режим доступу: https://pandas.pydata.org/

17. Python Software Foundation. *Official Python Documentation*. — https://docs.python.org/3/

18. DDoS-Attack Dataset from CAIDA [Електронний ресурс]. — Режим доступу: https://www.caida.org/data/passive/ddos/

19. ДУІКТ. Тези: Щерба А. А. *Аналіз DoS та DDoS атак із використанням Python-застосунку: підходи до виявлення та візуалізації* // Матеріали науково-практичної конференції. — К., 2024. — С. 108–109.

20. Khan M., Awan I. *Machine Learning in Detection of DoS Attacks: A Survey* // *IEEE Access*. — 2020. — Vol. 8. — P. 3456–3471.

Додатки

```
 import sys
import requests
import ipaddress
import time
from PyQt5 import QtWidgets, QtGui, QtCore, QtWebEngineWidgets
from scapy.all import rdpcap, TCP, UDP, ICMP, Raw
from collections import defaultdict, Counter
import csv
import matplotlib.pyplot as plt
from matplotlib.backends.backend_qt5agg import FigureCanvasQTAgg as FigureCanvas
import datetime
from fpdf import FPDF
import folium
import os


class PacketAnalyzerThread(QtCore.QThread):
   progress = QtCore.pyqtSignal(int)
   estimate = QtCore.pyqtSignal(str)
   finished = QtCore.pyqtSignal(dict, dict, dict)


   def __init__(self, file_path):
      super().__init__()
      self.file_path = file_path
```

```
  def get_country_by_ip(self, ip):
    try:
      if ipaddress.ip_address(ip).is_private:
        return ("Local", None, None)

      headers = {'User-Agent': 'Mozilla/5.0'}

      try:
        response = requests.get(f"http://ip-api.com/json/{ip}?fields=status,country,lat,lon",
headers=headers, timeout=3)
        if response.status_code == 200:
          data = response.json()
          if data.get("status") == "success":
            return (data.get("country", "Unknown"), data.get("lat"), data.get("lon"))
      except Exception as e:
        print(f"ip-api.com failed: {e}")

      try:
        response = requests.get(f"https://ipwho.is/{ip}", headers=headers, timeout=3)
        data = response.json()
        if data.get("success"):
          return (data.get("country", "Unknown"), data.get("latitude"), data.get("longitude"))
      except Exception as e:
        print(f"ipwho.is failed: {e}")
```

```
    except Exception as e:
        print(f"Unexpected error for IP {ip}: {e}")

    return ("Unknown", None, None)


def run(self):
    ip_cache = {}
    packets = rdpcap(self.file_path)
    traffic_data = {}
    time_series = {
        'syn': defaultdict(int),
        'udp': defaultdict(int),
        'icmp': defaultdict(int),
        'http': defaultdict(int)
    }
    ip_locations = {}
    total = len(packets)
    start_time = time.time()

    for index, pkt in enumerate(packets):
        if pkt.haslayer("IP"):
            ip = pkt["IP"].src
```

```
pkt_time = int(pkt.time)


if ip not in traffic_data:
    if ip in ip_cache:
        country, lat, lon = ip_cache[ip]
    else:
        country, lat, lon = self.get_country_by_ip(ip)
        ip_cache[ip] = (country, lat, lon)


    traffic_data[ip] = {
        'syn': 0, 'udp': 0, 'icmp': 0,
        'http': 0, 'dns_amp': 0,
        'first': pkt.time, 'last': pkt.time,
        'country': country
    }
    if lat and lon:
        ip_locations[ip] = {"country": country, "lat": lat, "lon": lon}
else:
    traffic_data[ip]['last'] = pkt.time
    if ip not in ip_cache:
        country, lat, lon = self.get_country_by_ip(ip)
        ip_cache[ip] = (country, lat, lon)
    traffic_data[ip]['country'] = ip_cache[ip][0]
```

```
    if pkt.haslayer(TCP) and pkt[TCP].flags == 'S':
        traffic_data[ip]['syn'] += 1
        time_series['syn'][pkt_time] += 1
    elif pkt.haslayer(UDP):
        traffic_data[ip]['udp'] += 1
        time_series['udp'][pkt_time] += 1
        if pkt.haslayer(Raw) and len(pkt[Raw].load) > 300:
            traffic_data[ip]['dns_amp'] += 1
    elif pkt.haslayer(ICMP):
        traffic_data[ip]['icmp'] += 1
        time_series['icmp'][pkt_time] += 1
    if pkt.haslayer(Raw) and b"HTTP" in pkt[Raw].load:
        traffic_data[ip]['http'] += 1
        time_series['http'][pkt_time] += 1


if index % 100 == 0 and index > 0:
    elapsed = time.time() - start_time
    rate = elapsed / index
    remaining = total - index
    estimate = remaining * rate
    self.estimate.emit(f"Estimated time left: {int(estimate)}s")
    self.progress.emit(int(index / total * 100))
```

```
        self.progress.emit(100)
        self.estimate.emit("Analysis complete.")
        self.finished.emit(traffic_data, time_series, ip_locations)

class TrafficAnalyzer(QtWidgets.QTabWidget):
    def __init__(self):
        super().__init__()
        self.setWindowTitle("DoS/DDoS Traffic Analyzer")
        self.setGeometry(100, 100, 1200, 800)

        self.setStyleSheet("""
            QTabBar::tab {
                background: #2e2e2e;
                color: #cccccc;
                padding: 6px;
                font-weight: bold;
                border: 1px solid #5c5c5c;
            }
            QTabBar::tab:selected {
                background: #3d3d3d;
                color: white;
            }
```

```
QWidget {
    background-color: #1e1e1e;
    color: white;
    font-family: Segoe UI;
    font-size: 10pt;
}
QPushButton {
    background-color: #444;
    border: 1px solid #666;
    border-radius: 4px;
    padding: 6px;
    color: white;
}
QPushButton:hover {
    background-color: #555;
}
QHeaderView::section {
    background-color: #3c3f41;
    color: white;
}
QTableWidget {
    gridline-color: #5c5c5c;
}
```

```
    """)

    self.init_ui()

def init_ui(self):
    self.tabs = QtWidgets.QTabWidget()
    self.setLayout(QtWidgets.QVBoxLayout())
    self.layout().addWidget(self.tabs)

    self.table_tab = QtWidgets.QWidget()
    self.map_tab = QtWidgets.QWidget()
    self.chart_tab = QtWidgets.QWidget()

    self.tabs.addTab(self.table_tab, "Traffic Table")
    self.tabs.addTab(self.map_tab, "IP Map")
    self.tabs.addTab(self.chart_tab, "Country Chart")

    self.init_table_tab()
    self.init_map_tab()
    self.init_chart_tab()

def init_table_tab(self):
    layout = QtWidgets.QVBoxLayout()
```

```
    self.load_button = QtWidgets.QPushButton("Load PCAP File")
    self.progress_bar = QtWidgets.QProgressBar()
    self.estimate_label = QtWidgets.QLabel()
    self.estimate_label.setStyleSheet("color: yellow")
    self.result_table = QtWidgets.QTableWidget()
    self.result_table.setColumnCount(9)
    self.result_table.setHorizontalHeaderLabels([
        "Source IP", "SYN", "UDP", "ICMP", "HTTP", "DNS AMP", "First", "Last", "Country"
    ])
    self.result_table.horizontalHeader().setStretchLastSection(True)

    layout.addWidget(self.load_button)
    layout.addWidget(self.progress_bar)
    layout.addWidget(self.estimate_label)
    layout.addWidget(self.result_table)
    self.table_tab.setLayout(layout)

    self.load_button.clicked.connect(self.load_pcap)

def init_map_tab(self):
    layout = QtWidgets.QVBoxLayout()
    self.map_view = QtWebEngineWidgets.QWebEngineView()
    layout.addWidget(self.map_view)
```

```
        self.map_tab.setLayout(layout)

    def init_chart_tab(self):
        layout = QtWidgets.QVBoxLayout()
        self.chart_canvas = FigureCanvas(plt.Figure())
        layout.addWidget(self.chart_canvas)
        self.chart_tab.setLayout(layout)

    def load_pcap(self):
        file_name, _ = QtWidgets.QFileDialog.getOpenFileName(self, "Open PCAP file", "", "PCAP
Files (*.pcap *.pcapng)")
        if file_name:
            self.analyzer_thread = PacketAnalyzerThread(file_name)
            self.analyzer_thread.progress.connect(self.progress_bar.setValue)
            self.analyzer_thread.estimate.connect(self.estimate_label.setText)
            self.analyzer_thread.finished.connect(self.update_results)
            self.analyzer_thread.start()

    def update_results(self, traffic_data, time_series, ip_locations):
        self.result_table.setRowCount(len(traffic_data))
        country_count = Counter()

        for row, (ip, data) in enumerate(traffic_data.items()):
            self.result_table.setItem(row, 0, QtWidgets.QTableWidgetItem(ip))
```

```
        self.result_table.setItem(row, 1, QtWidgets.QTableWidgetItem(str(data["syn"])))
        self.result_table.setItem(row, 2, QtWidgets.QTableWidgetItem(str(data["udp"])))
        self.result_table.setItem(row, 3, QtWidgets.QTableWidgetItem(str(data["icmp"])))
        self.result_table.setItem(row, 4, QtWidgets.QTableWidgetItem(str(data["http"])))
        self.result_table.setItem(row, 5, QtWidgets.QTableWidgetItem(str(data["dns_amp"])))
        self.result_table.setItem(row, 6,
QtWidgets.QTableWidgetItem(QtCore.QDateTime.fromSecsSinceEpoch(int(data["first"])).toStri
ng()))
        self.result_table.setItem(row, 7,
QtWidgets.QTableWidgetItem(QtCore.QDateTime.fromSecsSinceEpoch(int(data["last"])).toStri
ng()))
        self.result_table.setItem(row, 8, QtWidgets.QTableWidgetItem(data["country"]))

        country_count[data["country"]] += 1

    fig = self.chart_canvas.figure
    fig.clear()
    ax = fig.add_subplot(111)
    ax.bar(country_count.keys(), country_count.values(), color='skyblue')
    ax.set_title("IP Count per Country")
    ax.set_xlabel("Country")
    ax.set_ylabel("Count")
    ax.tick_params(axis='x', rotation=45, labelsize=8)
    fig.tight_layout()
```

```
        self.chart_canvas.draw()


        if ip_locations:
            fmap = folium.Map(location=[48.3794, 31.1656], zoom_start=3)
            for ip, info in ip_locations.items():
                folium.Marker(
                    location=[info["lat"], info["lon"]],
                    popup=f"{ip} ({info['country']})",
                    icon=folium.Icon(color="blue", icon="info-sign")
                ).add_to(fmap)
            map_file = "ip_map.html"
            fmap.save(map_file)
            self.map_view.setUrl(QtCore.QUrl.fromLocalFile(os.path.abspath(map_file)))


if __name__ == "__main__":
    app = QtWidgets.QApplication(sys.argv)
    window = TrafficAnalyzer()
    window.show()
    sys.exit(app.exec_())
```

Додаток на мові Python,для аналізу Dos,Ddos атак

Виконав:студент групи КНД-42
Щерба Артем

Слайд 1

Що таке атака Dos

Мета атаки DoS

Атака DoS намагається перевантажити ресурс, наприклад, сервер, щоб знизити його доступність для користувачів.

Вплив на користувачів

Ця атака призводить до зниження доступності сервісів, що впливає на легітимних користувачів, які не можуть отримати доступ.

Типи атак DoS

Існує кілька типів атак DoS, які можуть використовувати різні методи для перевантаження системи.

Слайд 2

Що таке атака DDoS

Що таке DDoS?

Атака DDoS є формою відмови в обслуговуванні, яка використовує кілька зламаних систем для атаки на ціль.

Потужність атаки

DDoS атаки є більш потужними, оскільки здійснюються з багатьох контрольованих джерел одночасно, ускладнюючи їх відбиття.

Складність відбиття

Відбиття DDoS атак є складнішим завданням через їх координованість і обсяги трафіку, які вони можуть генерувати.

Слайд 3

Основні відмінності між Dos і DDoS

Джерела атак

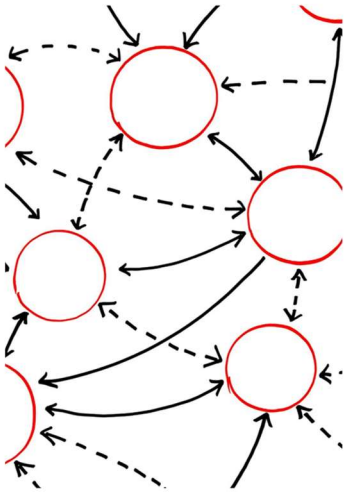
DoS атаки зазвичай походять з одного джерела, тоді як DDoS атаки мають безліч джерел. Це робить DDoS більш складним для виявлення.

Складність виявлення

DDoS атаки ускладнюють виявлення і відповідь на загрозу через їх розподілену природу. Це потребує більш складних методів захисту.

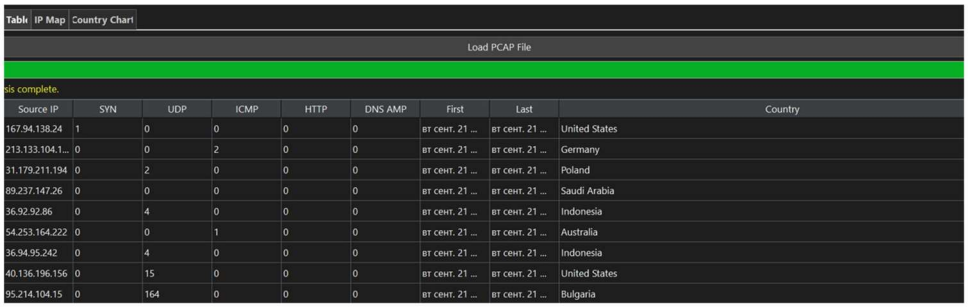
Масштаб атак

DDoS атаки зазвичай більш масштабні та ефективні, ніж DoS через велику кількість джерел. Це призводить до більш серйозних наслідків.



Слайд 4

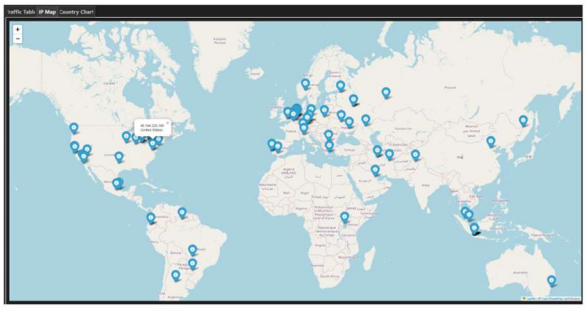
SYN UDP ICMP HTTP DNS Amplification



Source IP	SYN	UDP	ICMP	HTTP	DNS AMP	First	Last	Country
167.94.138.24	1	0	0	0	0	вт сент. 21 ...	вт сент. 21 ...	United States
213.133.104.1...	0	0	2	0	0	вт сент. 21 ...	вт сент. 21 ...	Germany
31.179.211.194	0	2	0	0	0	вт сент. 21 ...	вт сент. 21 ...	Poland
89.237.147.26	0	0	0	0	0	вт сент. 21 ...	вт сент. 21 ...	Saudi Arabia
36.92.92.86	0	4	0	0	0	вт сент. 21 ...	вт сент. 21 ...	Indonesia
54.253.164.222	0	0	1	0	0	вт сент. 21 ...	вт сент. 21 ...	Australia
36.94.95.242	0	4	0	0	0	вт сент. 21 ...	вт сент. 21 ...	Indonesia
40.136.196.156	0	15	0	0	0	вт сент. 21 ...	вт сент. 21 ...	United States
95.214.104.15	0	164	0	0	0	вт сент. 21 ...	вт сент. 21 ...	Bulgaria

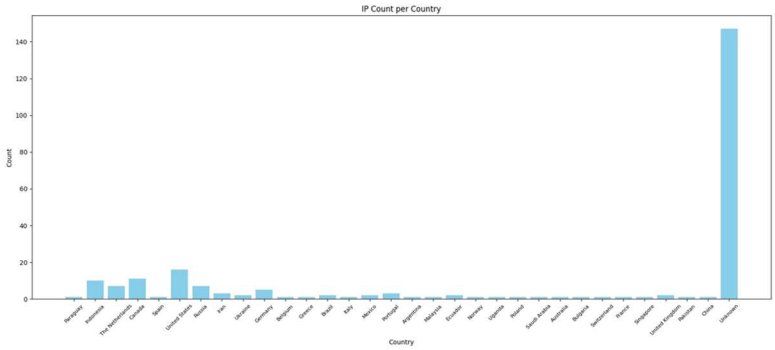
Слайд 5

IP-MAP



Слайд 6

Гістограма кількісті IP-адрес за країнами



Слайд 7

Слайд 8