Міністерство освіти і науки України

Національний технічний університет
«Дніпровська політехніка»

Навчально-науковий
інститут електроенергетики
(інститут)

Факультет інформаційних технологій
(факультет)
Кафедра інформаційних технологій та комп'ютерної інженерії
(повна назва)

# ПОЯСНЮВАЛЬНА ЗАПИСКА
## кваліфікаційної роботи ступеня бакалавра

**здобувача** Мусієнко Олександр Вікторович
(ПІБ)

**академічної групи** 123-22ск-1
(шифр)

**спеціальності** 123 Комп'ютерна інженерія
(код і назва спеціальності)

**за освітньо-професійною програмою** 123 Комп'ютерна інженерія
(офіційна назва)

**на тему** " Прикладне програмне забезпечення для моніторингу та виявлення DDoS-атак у мережевому трафіку комп'ютерних систем"
(назва за наказом ректора)

| Керівники | Прізвище, ініціали | Оцінка за шкалою | | Підпис |
|---|---|---|---|---|
| | | рейтинговою | інституційною | |
| кваліфікаційної роботи | доц. Каштан В.Ю. | | | |
| загального розділу | доц. Каштан В.Ю. | | | |
| спеціального розділу | доц. Каштан В.Ю. | | | |
| **Рецензент** | | | | |
| **Нормоконтролер** | проф. Цвіркун Л.І. | | | |

**Дніпро**
**2025**

**ЗАТВЕРДЖЕНО:**
завідувач кафедри
інформаційних технологій
та комп'ютерної інженерії
(повна назва)
_____ Гнатушенко В.В.
(підпис) (прізвище, ініціали)

"____" _____ 2025 року

# ЗАВДАННЯ
## на кваліфікаційну роботу
## ступеня бакалавр

**здобувача** Мусієнко О.В. **академічної групи** 123-22ск-1
(прізвище та ініціали) (шифр)

**спеціальності** 123 Комп'ютерна інженерія

**за освітньо-професійною програмою** Комп'ютерна інженерія
(офіційна назва)

**на тему** " Прикладне програмне забезпечення для моніторингу та виявлення DDoS-атак у мережевому трафіку комп'ютерних систем "

затверджену наказом ректора НТУ «Дніпровська політехніка» від 05.05.2025 № 336-с

| Розділ | Зміст | Термін виконання |
|---|---|---|
| Загальний розділ | На основі матеріалів виробничих практик, інших науково-технічних джерел показати актуальність завдання, сформулювати мету та задачі виконання кваліфікаційної роботи | 10.02.2025 |
| Спеціальний розділ | Сформулювати найменування й призначення програмного забезпечення для моніторингу та виявлення DDoS-атак у мережевому трафіку, висунути технічні вимоги до нього | 15.03.2025 |
| | Реалізувати програмний модуль, використовуючи відповідні інструменти програмування та враховуючи принципи розробки надійних та ефективних програмних систем, що є частиною комп'ютерної інженерії | 07.05.2025 |
| | Протестувати розроблене програмного забезпечення для моніторингу та виявлення DDoS-атак у мережевому трафіку на предмет її функціональності, продуктивності обміну даними в реальному часі та стійкості до можливих збоїв | 31.05.2025 |

**Завдання видано** _____ доц. Каштан В.Ю.
(підпис керівника) (прізвище, ініціали)

**Дата видачі** 25.01.2025

**Дата подання до екзаменаційної комісії** _____

**Прийнято до виконання** _____ Мусієнко О.В.
(підпис студента) (прізвище, ініціали)

# РЕФЕРАТ

Пояснювальна записка: 69 с., 19 рис., 1 табл., 1 додаток, 14 джерел.

Об’єкт дослідження – процеси моніторингу та виявлення розподілених атак типу «відмова в обслуговуванні» (DDoS) у мережевому трафіку комп’ютерних систем.

Предмет дослідження – методи, алгоритми та програмні засоби реалізації моніторингу мережевого трафіку і виявлення DDoS-атак із застосуванням прикладного програмного забезпечення.

Мета роботи – розробити прикладне програмне забезпечення для моніторингу та виявлення DDoS-атак у мережевому трафіку комп’ютерних систем з метою підвищення ефективності реагування на загрози інформаційної безпеки.

Завдання дослідження:

1. Провести аналіз предметної області, зокрема вивчити принципи реалізації DDoS-атак, їх класифікацію та сучасні методи їхнього виявлення.

2. Проаналізувати наявні програмні рішення для моніторингу та виявлення DDoS-атак, визначити їхні функціональні особливості, переваги та недоліки.

3. Спроєктувати архітектуру прикладного програмного забезпечення для моніторингу та виявлення аномальної активності, притаманної DDoS-атакам.

4. Реалізувати програмний прототип із модулями збору та аналізу мережевого трафіку, виявлення аномалій та візуалізації результатів для користувача.

5. Провести тестування створеного програмного забезпечення на тестовому або реальному мережевому трафіку для оцінки його ефективності у виявленні DDoS-атак.

Ключові слова: DDoS-атака, мережевий трафік, моніторинг, аналіз трафіку, розподілені атаки.

# ЗМІСТ

# ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАК, ОДИНИЦЬ ТА ТЕРМІНІВ

DDoS – Distributed Denial of Service

IoT – Інтернет речей

НМ – нейронні мережі

IP – Internet Protocol

TCP – Transmission Control Protocol

UDP – User Datagram Protocol

ICMP – Internet Control Message Protocol

# ВСТУП

Всі сфери життєдіяльності людини тісно пов’язані з інформаційними технологіями та цифровими інструментами. Комп’ютерні системи, мобільні пристрої та глобальна мережа Інтернет щоденно обробляють, зберігають і передають великі обсяги конфіденційної інформації – як персонального характеру, так і корпоративного значення, що підтримує функціонування численних приватних і державних структур. У зв’язку з цим інформаційна безпека стає одним із ключових факторів стабільного функціонування цифрового суспільства.

Кібератаки, визначаючись як скоординовані дії, спрямовані на несанкціонований доступ, крадіжку інформації або завдання шкоди комп'ютерним системам та мережам [1], становлять серйозну загрозу для окремих осіб та суспільства в цілому.

Серед різноманіття кіберзагроз особливе місце займають розподілені атаки типу «відмова в обслуговуванні» (DDoS-атаки). Ці скоординовані атаки, спрямовані на виведення з ладу цільових ресурсів шляхом їх перевантаження численними запитами, набули значного поширення та можуть призвести до значних фінансових втрат, порушення функціонування критичної інфраструктури та підриву довіри користувачів.

Актуальність теми дослідження зумовлена стрімким зростанням кількості кібератак, зокрема DDoS-атак, протягом останніх років. Статистичні дані за 2023-2024 роки наочно демонструють цю тенденцію. Так, за даними щоквартального звіту «Лабораторії Касперського», було зафіксовано близько 57 тисяч DDoS-атак. Крім того, компанія Cloudflare повідомляє про вражаюче зростання кількості DDoS-атак з метою викупу на 67% у 2023 році [2]. Ці цифри підкреслюють нагальну потребу в розробці та впровадженні ефективних засобів моніторингу та виявлення DDoS-атак для забезпечення стабільної та безпечної роботи комп’ютерних систем.

Виходячи з вищезазначеного, метою даної кваліфікаційної роботи є розробка прикладного програмного забезпечення для моніторингу та виявлення DDoS-атак у мережевому трафіку комп’ютерних систем.

# 1 ЗАГАЛЬНИЙ РОЗДІЛ

## 1.1 Аналіз предметної області

Розподілена атака типу «відмова в обслуговуванні» (DDoS-атака) являє собою зловмисну спробу порушити штатний обмін даними з цільовим сервером, онлайн-сервісом або мережевою інфраструктурою шляхом її перевантаження надмірним потоком інтернет-трафіку [3]. Тенденції 2023 та 2024 років демонструють значне зростання як частоти, так і складності DDoS-атак. За даними різних звітів, кількість DDoS-атак у 2024 році зросла на вражаючі 82% порівняно з 2023 роком, а загальний обсяг атакуючого трафіку майже подвоївся [1, 2]. У свою чергу, виявлення DDoS-атак є критично важливим процесом, спрямованим на ідентифікацію та відокремлення шкідливого розподіленого трафіку від легітимного з метою забезпечення ефективного реагування та нейтралізації загроз.

Принцип функціонування DDoS-атак [3] базується на використанні мереж інфікованих комп'ютерів, підключених до Інтернету. Ці мережі складаються з великої кількості комп'ютерів та інших пристроїв, які були заражені шкідливим програмним забезпеченням, що надає зловмиснику можливість дистанційного керування ними. Однак, у 2023-2024 роках спостерігається зростання потужності атак, які сягають рекордних значень у кілька терабіт на секунду, що робить традиційні методи захисту менш ефективними [2, 3]. Збільшується також кількість атак, спрямованих на конкретні додатки та API, що вимагає більш глибокого аналізу трафіку на рівні додатків для їх виявлення та фільтрації.

## 1.2 Типи DDoS-атак

DDoS-атаки, спрямовані на порушення нормального функціонування онлайн-сервісів, можуть бути класифіковані за кількома ключовими критеріями, що відображають їх архітектуру, структуру та загальну поведінку. Однією з фундаментальних категорій є рівень комп'ютеризації, що визначає ступінь автоматизації процесу атаки. Ця категорія поділяється на три підтипи (рис.1.1).

Атаки на основі інструкцій є повністю ручними, охоплюючи кожен етап від виявлення потенційних цілей до безпосереднього використання ресурсів для ініціації атаки. Другим підтипом є напівзаздалегідь налаштовані атаки, які поєднують ручні та автоматизовані механізми. У цьому випадку зловмисник автоматизує всі кроки, окрім фактичної команди для початку атаки. Нарешті, попередньо встановлені атаки є повністю автоматизованими, не потребуючи ручного втручання [6].

Друга важлива категорія – мережа атаки – описує зв'язок між ресурсами, що використовуються для здійснення атаки, та джерелом інструкцій, що її ініціює. У контексті ботнетів, які часто застосовуються для DDoS-атак, класифікація залежить від способу передачі команд від ботмайстра до окремих ботів.

Наступною категорією є пригнічені вразливості, яка описує фактичний механізм атаки. Ця категорія включає чотири основні підтипи. Атаки повені (flood attacks) характеризуються перевантаженням цілі надмірним трафіком з метою спричинення відмови в обслуговуванні. Атаки інтенсифікації (amplification attacks) є більш складним варіантом атак повені, що використовують певний механізм для збільшення обсягу трафіку, який зловмисник надсилає жертві. Атаки з використанням протоколу (protocol exploitation attacks) є більш підступною формою атаки, що експлуатує вразливості в мережевих протоколах, які використовує цільова система. Нарешті, аналіз шкідливих пакетів (malformed packet attacks) передбачає створення та надсилання неправильно сформованого мережевого трафіку жертві, що може призвести до збоїв або вичерпання ресурсів при спробі його обробки.

Категорія впливу класифікує DDoS-атаки на основі наслідків атаки, поділяючи їх на хаотичні (disruptive) та принизливі (degradative). Хаотична атака призводить до повного припинення надання послуги, тоді як принизлива атака спричиняє лише часткове погіршення якості обслуговування.

Остання категорія, динаміка інтенсивності атаки, враховує зміну розміру атаки в часі. Підкатегорія безперервної інтенсивності атаки (constant rate attacks) характеризується постійним потоком трафіку, спрямованим на жертву. Натомість, атаки зі змінною інтенсивністю (variable rate attacks) включають періодичні

коливання інтенсивності атакуючого трафіку. Ці різноманітні категорії та підкатегорії допомагають краще розуміти природу та механізми DDoS-атак, що є важливим для розробки ефективних стратегій захисту [6].

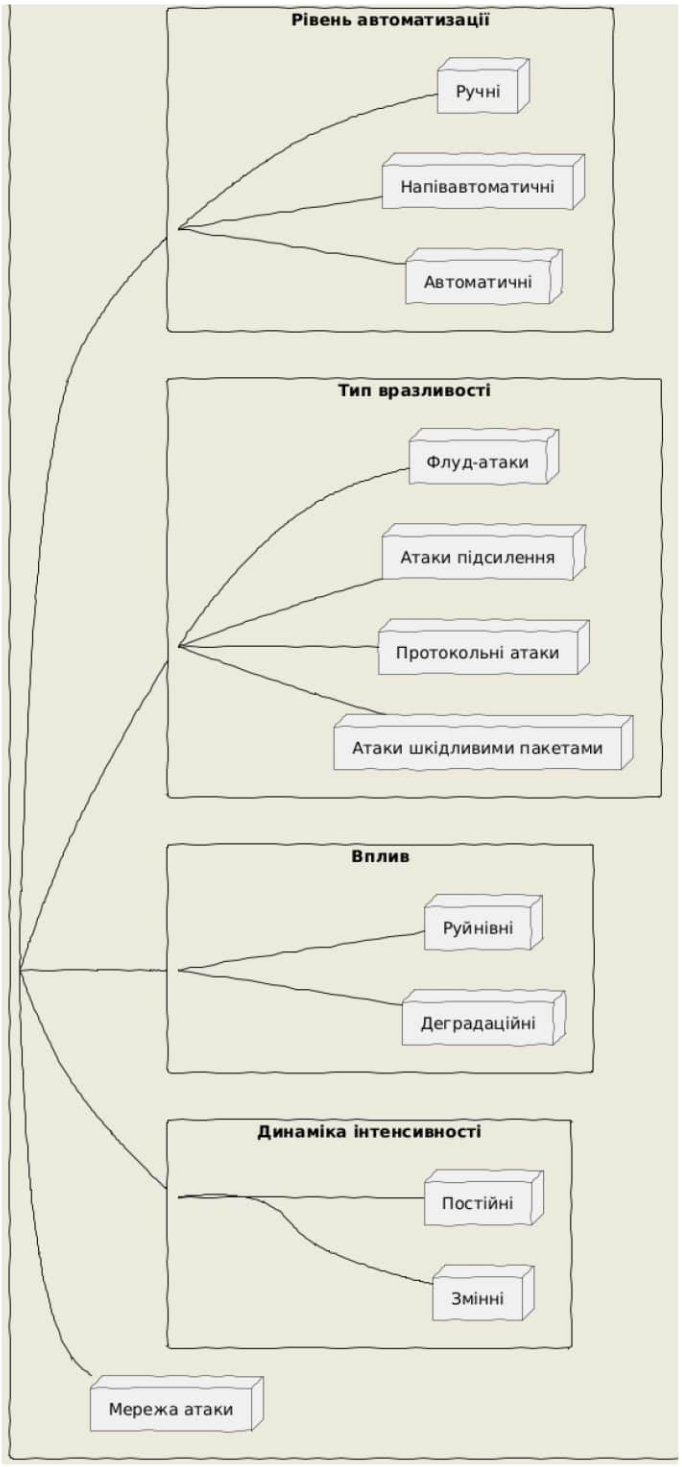



Рисунок 1.1 – Класифікація DDoS-атак

### 1.1.1 Типи DDoS-атак за рівнями мережевої моделі OSI

DDoS-атаки можуть бути класифіковані залежно від того, на який рівень мережевої взаємодії вони спрямовані [8]. Такий підхід дозволяє краще зрозуміти механізми їх дії та розробляти ефективні стратегії захисту. За цією класифікацією виділяють три основні типи DDoS-атак: об'ємні, протокольні та прикладного рівня.

Об'ємні DDoS-атаки (Network Layer Attacks - Рівень 3 OSI) націлені на вичерпання пропускної здатності мережевої інфраструктури жертви шляхом надсилання величезного обсягу шкідливого трафіку. Основна мета полягає в тому, щоб перевантажити канали зв'язку та зробити цільовий ресурс недоступним для легітимних користувачів. Часто для збільшення ефективності таких атак зловмисники використовують методи підсилення, такі як DNS-посилення або NTP-посилення, коли запити, відправлені на спеціально налаштовані сервери, призводять до значно більших відповідей, спрямованих на жертву [7].

Крім того, для генерації великих обсягів трафіку активно використовуються ботнети, які можуть складатися з великої кількості скомпрометованих пристроїв, включаючи пристрої Інтернету речей (IoT). Оскільки багато IoT-пристроїв мають слабкі системи безпеки, вони стають легкою мішенню для зловмисників, які використовують їх для організації масштабних атак.

До основних видів об'ємних DDoS-атак належать [7]:

– SYN-флуд – атака, що експлуатує процес встановлення TCP-з'єднання, відправляючи велику кількість SYN-пакетів без подальшого підтвердження, що призводить до переповнення черги підключень на сервері;

– ICMP-флуд – затоплення цілі великою кількістю ICMP-пакетів (наприклад, ping-запитів), що може призвести до перевантаження мережевих пристроїв та вичерпання їх ресурсів;

– UDP-флуд – надсилання великої кількості UDP-пакетів на випадкові порти цільового сервера, що може призвести до перевантаження мережевої смуги пропускання та ресурсів обробки пакетів;

– Атаки з посиленням DNS (DNS Amplification) – відправлення невеликих запитів на DNS-сервери зі підробленою IP-адресою жертви, що призводить до

отримання великих відповідей, які спрямовуються на ціль, значно збільшуючи обсяг атакуючого трафіку;

– атаки з посиленням NTP (NTP Amplification) – аналогічно DNS-посиленню, використовуються NTP-сервери для генерації великих обсягів трафіку у відповідь на невеликі запити зі підробленою IP-адресою жертви.

Протокольні DDoS-атаки (Transport Layer - Рівень 4 та частково Application Layer - Рівень 7 OSI) націлені на експлуатацію особливостей роботи мережевих протоколів на транспортному (рівень 4) та частково прикладному (рівень 7) рівнях моделі OSI. Ці атаки часто використовують сценарії, в яких сервер ініціює обробку запиту та очікує подальшої взаємодії, виділяючи для цього певні ресурси. Велика кількість таких незавершених сесій або неправильно сформованих запитів може призвести до виснаження серверних ресурсів, уповільнення або повної зупинки обміну даними.

До основних видів протокольних DDoS-атак належать [3]:

– атаки типу «людина посередині» (Man-in-the-Middle - MitM) – хоча це не є класичною атакою на відмову в обслуговуванні, зловмисник перехоплює та, можливо, модифікує обмін даними між двома сторонами, що може призвести до порушення зв'язку та конфіденційності;

– підміна DNS (DNS Spoofing) – зловмисник підміняє DNS-записи, перенаправляючи трафік на шкідливі сервери, що може призвести до недоступності легітимних ресурсів;

– атаки SYN Flood (Згадувалося також в об'ємних атаках, оскільки може призводити як до перевантаження каналу, так і до вичерпання ресурсів сервера, що підтримують незавершені з'єднання);

– атаки смурфів (Smurf Attacks) використовують ICMP-пакети, відправлені на широкомовні адреси з підробленою IP-адресою жертви, що призводить до масової відповіді від усіх хостів у мережі на адресу жертви;

– злом TCP/IP (TCP/IP Hijacking) – зловмисник перехоплює активне TCP-з'єднання між двома хостами та вставляє свій шкідливий трафік або перериває сеанс;

– IP-спуфінг (IP Spoofing) – підробка IP-адреси відправника в мережевих

пакетах, що може ускладнити відстеження джерела атаки та використовуватися для атак підсилення;

– отруєння ARP (ARP Poisoning) – зловмисник надсилає підроблені ARP-повідомлення в локальній мережі, асоціюючи свою MAC-адресу з IP-адресою іншого пристрою (наприклад, шлюзу), що дозволяє йому перехоплювати трафік.

DDoS-атаки прикладного рівня (Application Layer Attacks - Рівень 7 OSI) спрямовані безпосередньо на програмне забезпечення та сервіси, що працюють на веб-серверах та інших онлайн-платформах. Ці атаки часто імітують легітимні запити користувачів, що робить їх складнішими для виявлення та блокування, оскільки вони можуть не відрізнятися за форматом від звичайного трафіку. Метою таких атак може бути крадіжка даних, порушення бізнес-процесів, виведення з ладу веб-сайтів або вимагання коштів.

Важливо зазначити, що DDoS-атаки прикладного рівня можуть бути частиною багатовекторних атак, комбінуючись з об'ємними та протокольними атаками для підвищення шансів на успішне виведення сервісу з ладу.

До основних видів DDoS-атак прикладного рівня належать [2]:

– впровадження шкідливого SQL-коду в запити до бази даних, що може дозволити зловмиснику отримати несанкціонований доступ до даних, змінити їх або видалити. Хоча це частіше розглядається як атака на безпеку, велика кількість таких запитів може призвести до перевантаження бази даних та відмови в обслуговуванні;

– впровадження шкідливого JavaScript-коду на веб-сторінки, який виконується в браузерах інших користувачів, що може призвести до крадіжки сесійних даних, перенаправлення на шкідливі сайти або інших зловмисних дій. Велика кількість таких запитів може перевантажити веб-сервер;

– експлуатація вразливостей у веб-додатках, що дозволяють зловмиснику включати віддалені файли зі шкідливим кодом, який може бути виконаний на сервері, що може призвести до порушення роботи сервісу;

– HTTP-флуд веб-сервера великою кількістю HTTP-запитів (GET або POST), що може призвести до вичерпання його ресурсів (CPU, пам'ять, з'єднання) та відмови в обслуговуванні легітимних користувачів. Ці запити часто імітують дії

звичайних браузерів, що ускладнює їх фільтрацію.

Об'ємні DDoS-атаки є одним з основних типів розподілених атак на відмову в обслуговуванні, які націлені на мережевий рівень (рівень 3 моделі OSI). Їхньою головною метою є перевантаження пропускної здатності мережі та інфраструктури жертви шляхом створення величезного обсягу шкідливого трафіку (табл.1.1).

Таблиця 1.1 – Об'ємні DDoS-атаки

| Тип | Метрика | Рівень OSI | Особливості | Приклади |
|---|---|---|---|---|
| Об'ємний | Гбіт/с, біт/с | Мережевий (рівень 3) | Великий об'єм трафіку, використання ботів та ботнетів IoT, методи посилення | ICMP-флуд, UDP-флуд, DNS-ампліфікація, NTP-ампліфікація |
| Протокольний | Кількість запитів/пакетів за сесію, виснаження ресурсів сервера | Транспортний (рівень 4), Прикладний (рівень 7) | Експлуатація особливостей протоколів, встановлення великої кількості незавершених сесій, неправильно сформовані пакети, виснаження ресурсів | Атаки типу «людина посередині» (MitM), підміна DNS, SYN Flood, атаки смурфів, злом TCP/IP, IP-спуфінг, отруєння ARP |
| Прикладний | Кількість запитів/сесій на додаток | Прикладний (рівень 7) | Імітація легітимних запитів, націленість на конкретні функції та вразливості додатків, складність виявлення та блокування | SQL-ін'єкція, міжсайтовий скриптинг (XSS), віддалене включення файлів (RFI) |

Концепція об'ємної атаки полягає у тому, щоб направити на цільовий ресурс максимально можливий обсяг трафіку, що перевищує його здатність обробляти запити. Для досягнення цієї мети зловмисники часто використовують методи підсилення, такі як DNS-посилення та NTP-посилення, які дозволяють генерувати значно більший обсяг трафіку у відповідь на відносно невеликі запити.

Важливу роль у проведенні об'ємних атак відіграють ботнети, що складаються з великої кількості скомпрометованих пристроїв, включаючи пристрої Інтернету речей (IoT). Недостатній рівень безпеки багатьох IoT-пристроїв робить їх легкою мішенню для зловмисників, які використовують їх для організації масштабних атак, генеруючи значні обсяги трафіку.

До типових прикладів об'ємних DDoS-атак належать: SYN-флуд, ICMP-флуд, UDP-флуд, атаки з посиленням DNS та атаки з посиленням NTP.

### 1.1.2 Мережеве підключення та передача даних

У процесі передачі даних через мережу інформація розбивається на окремі блоки, які називаються пакетами. Пакет є основною одиницею даних на мережевому рівні (рівень 3) моделі OSI. Кожен пакет містить не лише частину передаваних даних, але й службову інформацію, необхідну для його маршрутизації та доставки до кінцевого отримувача.

Мережеве підключення характеризується різноманітністю смуг пропускання, маршрутів передачі та великою кількістю підключених пристроїв, які можуть обмінюватися даними незалежно один від одного. Така децентралізована структура забезпечує гнучкість та стійкість мережі. У випадку втрати або пошкодження окремих пакетів, їх повторна передача може бути здійснена без необхідності повторного надсилання всього обсягу даних.

Більшість мережевих пакетів складаються з трьох основних частин, кожна з яких містить певний набір атрибутів (як показано на рисунку 1.2, який, на жаль, мені недоступний для візуалізації). Ці частини зазвичай включають:

– заголовок (Header) містить службову інформацію, необхідну для

маршрутизації пакета, таку як адреси відправника та отримувача (IP-адреси), інформацію про протокол, порядковий номер пакета (для відновлення послідовності), час життя пакета (TTL) та інші службові поля;

– дані (Payload) – власне корисне навантаження пакета, тобто фрагмент передаваної інформації. Розмір цієї частини може варіюватися залежно від протоколу та налаштувань мережі;

– хвіст (Trailer) може містити додаткову службову інформацію, наприклад, контрольну суму (checksum) для перевірки цілісності даних під час передачі.

Розбиття даних на пакети та їх незалежна передача є ключовим принципом функціонування сучасних комп'ютерних мереж, що забезпечує ефективність, надійність та гнучкість обміну інформацією.


Рисунок 1.2 – Структура мережевого пакету

### 1.3 Ознаки виявлення DDoS-атаки

Основними об'єктивними ознаками успішної DDoS-атаки є помітне уповільнення або повна недоступність цільових служб. Однак для більш раннього та точного виявлення атаки можна аналізувати ряд специфічних ознак [8], що базуються на структурі мережевих пакетів [7]:

– значне збільшення трафіку, що надходить з однієї або діапазону IP-адрес, може свідчити про скоординовану атаку;

– потік запитів від користувачів з ідентичними або дуже схожими характеристиками, такими як тип пристрою, геолокація або версія веб-браузера, може вказувати на використання ботнету;

– аномально висока кількість запитів, спрямованих на конкретну веб-сторінку або кінцеву точку API, може бути ознакою спроби перевантаження цього ресурсу;

– відхилення від звичайних моделей трафіку, такі як нерівномірні інтервали між запитами або нетипові розміри пакетів, можуть бути індикаторами атаки;

– існують й інші, менш очевидні характеристики мережевого трафіку, які при ретельному аналізі можуть вказувати на DDoS-активність.

Процес виявлення DDoS-атак часто включає аналіз аномалій мережевого трафіку. Це передбачає постійний моніторинг мережі та пошук відхилень від встановлених контрольних показників, що відображають нормальний режим роботи мережі.

Загалом, класифікація систем виявлення DDoS-атак схожа на класифікацію систем виявлення вторгнень. Основними підходами є виявлення на основі сигнатур та виявлення на основі аномалій.

Виявлення атак на основі сигнатур використовується для ідентифікації вже відомих типів атак. Для його застосування необхідна база даних з відомими сигнатурами атак – характерними патернами в мережевому трафіку, які однозначно вказують на певну атаку. Перевагою цього методу є висока точність виявлення відомих загроз. Однак для нових або модифікованих атак цей метод є

неефективним. Крім того, порівняння кожного пакета з великою базою сигнатур може вимагати значних обчислювальних ресурсів.

Виявлення атак на основі аномалій полягає у виявленні незвичайної активності в мережевому трафіку, яка відхиляється від визначеної "нормальної" поведінки системи або додатків. При виявленні таких відхилень генерується попередження про можливу аномалію. Основні труднощі цього методу полягають у точному визначенні типової поведінки мережі, виборі оптимальних порогів для спрацьовування сигналів тривоги та мінімізації кількості хибних спрацьовувань.

Існують також базові вбудовані рішення для виявлення DDoS-атак, які пропонуються деякими мережевими пристроями, такими як балансувальники навантаження, брандмауери та системи запобігання вторгненням (IPS). Однак ці рішення часто є універсальними та можуть бути недостатньо ефективними проти новітніх та складних типів атак.

На сьогоднішній день найбільш ефективним способом оперативного виявлення DDoS-атак є використання спеціалізованих методів, розроблених виключно для ідентифікації шкідливого трафіку DDoS-атак.

В умовах, коли зловмисники починають активно використовувати технології штучного інтелекту та машинного навчання для здійснення кібератак, застосування цих же технологій для раннього виявлення загроз стає логічним та перспективним рішенням.

Для ідентифікації незаконного трафіку необхідно глибоко дослідити характеристики легітимних мережевих пакетів, що генеруються різними додатками, та порівняти їх з характеристиками підроблених пакетів, які створюються інструментами зловмисників. Результати цього порівняння можуть бути використані як патерни для розрізнення трафіку та представлені як вхідні змінні для навчання нейронних мереж (НМ).

Навчання НМ може здійснюватися на основі шаблонів заголовків пакетів, що включають такі атрибути, як адреси джерел, ідентифікатори та порядкові номери, у поєднанні з номерами портів джерела та призначення. Аналіз цих характеристик дозволяє НМ виявляти аномалії, що можуть свідчити про DDoS-атаку.

## 1.4 Сучасні підходи виявлення DDoS-атак

У сучасній науковій літературі спостерігається активний розвиток методів виявлення та класифікації DDoS-атак [9,10], де ключову роль відіграє застосування моделей нейронних мереж з різноманітними архітектурами, такими як ANN, Random Forest (RF), Logistic Regression (LG), K-Nearest Neighbors (KNN), Naive Bayes (NB), Multi-Layer Perceptron (MLP) та Support Vector Machine (SVM). Кожен з цих підходів використовує різні принципи машинного навчання для аналізу мережевого трафіку та ідентифікації аномалій, що можуть свідчити про атаку.

Для ефективного аналізу та порівняння існуючих досліджень у цій галузі доцільно використовувати ряд критеріїв оцінки. Першим критерієм є мета дослідження, яка зазвичай полягає у виявленні присутності DDoS-атаки (бінарна класифікація) або у визначенні конкретного типу атаки (багатокласова класифікація). Наприклад, дослідження може бути спрямоване на розробку моделі MLP для виявлення HTTP-флуду шляхом аналізу характеристик HTTP-запитів, таких як частота запитів з однієї IP-адреси, типи запитів (GET, POST), розмір запитів та наявність специфічних заголовків. Іншим прикладом є розробка системи на основі RF для класифікації об'ємних атак, розрізняючи UDP-флуд за високою швидкістю надходження UDP-пакетів на випадні порти та DNS-ампліфікацію за характерними великими розмірами відповідей DNS-серверів на невеликі запити зі підробленими IP-адресами жертви.

Другим важливим критерієм є вхідні дані, що використовуються для навчання та тестування моделей ІНС. Тип та якість цих даних суттєво впливають на ефективність виявлення. Прикладом вхідних даних можуть бути заголовки IP-пакетів, що містять інформацію про IP-адреси джерела та призначення, порти, протоколи (TCP, UDP, ICMP) та прапори TCP. Дослідження, що використовує SVM для виявлення SYN-флуду, може навчати свою модель на основі статистики SYN-пакетів, таких як кількість SYN-пакетів, що надходять на певний порт за одиницю часу, та співвідношення SYN-пакетів до встановлених з'єднань. Інше дослідження може використовувати агреговані характеристики трафіку за певні часові вікна, наприклад, середню швидкість передачі пакетів, дисперсію розмірів пакетів або

кількість нових з'єднань, для виявлення аномальних патернів, характерних для DDoS-атак.

Третім критерієм є метод класифікації, тобто конкретний алгоритм машинного навчання та його конфігурація. Наприклад, дослідження може використовувати KNN з визначеною кількістю сусідів (k=5) та евклідовою відстанню для класифікації трафіку як нормального або аномального на основі схожості з попередньо розміченими даними. Інший приклад – застосування наївного баєсового класифікатора (NB), який на основі ймовірностей появи певних ознак (наприклад, певних значень полів у заголовках пакетів) визначає, чи належить даний трафік до класу атаки або нормального трафіку.

Нарешті, четвертим критерієм є результат класифікації, який оцінюється за допомогою різноманітних метрик. Наприклад, точність (accuracy) показує загальну частку правильно класифікованих зразків. Повнота (recall) відображає здатність моделі виявляти всі наявні випадки атаки. Точність (precision) показує частку справжніх атак серед усіх випадків, класифікованих як атака. F1-міра є гармонійним середнім точності та повноти. Дослідження, що пропонує нову модель ANN для виявлення DDoS-атак, повинно продемонструвати її ефективність, наводячи значення цих метрик на тестовому наборі даних та порівнюючи їх з результатами існуючих підходів. Важливими також є показники коефіцієнта хибних спрацьовувань (FPR), який показує частку нормального трафіку, помилково класифікованого як атака, та коефіцієнта істинно позитивних спрацьовувань (TPR), що відповідає повноті.

Аналіз сучасних підходів виявлення DDoS-атак на основі цих критеріїв дозволяє не лише оцінити їхні сильні та слабкі сторони, але й визначити перспективні напрямки для подальших досліджень, спрямованих на розробку більш ефективних та надійних систем захисту від цього типу кіберзагроз.

## 1.5 Програмні рішення, продукти та інструменти для виявлення DDoS-атак

Ринок кібербезпеки пропонує широкий спектр програмних рішень, продуктів та інструментів, розроблених для виявлення та пом'якшення DDoS-атак. Ці рішення варіюються за складністю, функціональністю, рівнем інтеграції та підходом до виявлення загроз.

Мережеві пристрої з вбудованими функціями виявлення DDoS [10, 11]:

- брандмауери нового покоління (NGFW);
- системи запобігання вторгненням (IPS);
- балансувальники навантаження.

Сучасні брандмауери часто включають базові функції виявлення аномалій трафіку та фільтрації шкідливих запитів, що можуть бути ознаками DDoS-атак. Вони можуть аналізувати стан з'єднань, контролювати швидкість трафіку та блокувати підозрілі IP-адреси. Брандмауери від Palo Alto Networks, Fortinet, Cisco Firepower мають можливості виявлення флуд-атак та інших аномалій мережевого рівня.

IPS аналізують мережевий трафік на предмет відомих сигнатур атак та аномальної поведінки. Вони можуть виявляти спроби експлуатації вразливостей, які можуть бути використані для організації DDoS-атак на прикладному рівні. Snort, Suricata є популярними IPS з правилами виявлення різноманітних атак, включаючи ті, що можуть бути використані в DDoS.

Балансувальники навантаження: Хоча основним їхнім завданням є розподіл трафіку між серверами, деякі балансувальники навантаження мають вбудовані механізми для виявлення та відхилення аномальних запитів, що можуть свідчити про DDoS. Вони можуть контролювати кількість з'єднань, швидкість запитів та інші метрики. F5 BIG-IP Local Traffic Manager (LTM) пропонує функціональність для виявлення та пом'якшення L7 DDoS-атак.

Спеціалізовані рішення для захисту від DDoS:

- локальні апаратні та програмні комплекси;
- хмарні сервіси захисту від DDoS.

Локальні апаратні та програмні комплекси – це спеціалізовані пристрої або програмне забезпечення, які встановлюються в мережі організації для аналізу вхідного та вихідного трафіку з метою виявлення та блокування DDoS-атак. Вони часто використовують комбінацію сигнатурного аналізу, поведінкового аналізу та методів машинного навчання. Приклад: Arbor Networks (тепер Netscout), Radware DefensePro є відомими рішеннями для локального захисту від DDoS.

Провайдери хмарних сервісів безпеки пропонують послуги з виявлення та пом'якшення DDoS-атак, перенаправляючи трафік через свої потужні інфраструктури для фільтрації шкідливих запитів перед тим, як вони досягнуть цільового сервера. Ці сервіси мають велику пропускну здатність та використовують різноманітні методи захисту. Приклад: Cloudflare, Akamai, AWS Shield, Google Cloud Armor є популярними хмарними сервісами захисту від DDoS.

Інструменти аналізу мережевого трафіку:

– системи моніторингу мережі;

– інструменти захоплення та аналізу пакетів;

– SIEM-системи (Security Information and Event Management).

Інструменти, які збирають та аналізують дані про мережевий трафік, дозволяючи адміністраторам виявляти аномалії та підозрілу активність, що може бути ознакою DDoS-атаки. Приклад: SolarWinds Network Performance Monitor (рис.1.3), PRTG Network Monitor можуть відображати обсяги трафіку, швидкість з'єднань та інші метрики, що допомагають виявити аномалії.

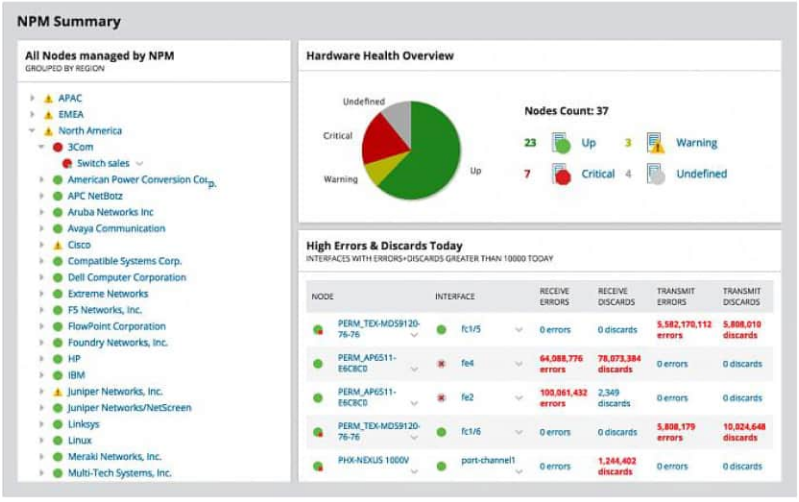



Рисунок 1.3 – SolarWinds Network Performance Monitor

Програми, які дозволяють захоплювати та детально аналізувати мережеві пакети, що може бути корисним для дослідження інцидентів безпеки та виявлення характеристик DDoS-атак. Приклад: Wireshark, tcpdump є потужними інструментами для аналізу мережевих пакетів.

SIEM-системи збирають та корелюють події безпеки з різних джерел, включаючи мережеві пристрої, сервери та програми, що може допомогти виявити скоординовані атаки, такі як DDoS. Приклад: Splunk, ELK Stack (Elasticsearch, Logstash, Kibana), IBM QRadar можуть використовуватися для аналізу логів та виявлення аномалій трафіку.

Деякі сучасні рішення для виявлення DDoS-атак використовують алгоритми машинного навчання для аналізу патернів трафіку та виявлення аномалій, які можуть не бути очевидними для традиційних методів на основі сигнатур або правил. Ці системи можуть навчатися на нормальному трафіку та виявляти відхилення, що свідчать про атаку. Деякі продукти від Netscout та Radware використовують поведінковий аналіз на основі машинного навчання для виявлення складних DDoS-атак.

Вибір конкретного рішення або набору інструментів залежить від розміру організації, її інфраструктури, рівня ризиків та бюджету. Комбінація різних підходів, включаючи як превентивні заходи (наприклад, використання CDN), так і інструменти для виявлення та реагування, є ключем до ефективного захисту від DDoS-атак.

### 1.6 Мета і задачі і роботи

Метою даної кваліфікаційної роботи є розробка прикладного програмного забезпечення для моніторингу та виявлення DDoS-атак у мережевому трафіку комп’ютерних систем.

Для досягнення поставленої мети необхідно вирішити наступні задачі:

– провести аналіз предметної області, включаючи дослідження принципів роботи DDoS-атак, їхніх типів та сучасних методів виявлення;

– вивчити існуючі програмні рішення, продукти та інструменти для моніторингу та виявлення DDoS-атак, визначити їхні переваги та недоліки;.

– розробити архітектуру прикладного програмного забезпечення для моніторингу та виявлення DDoS-атак.

– реалізувати програмний прототип, що включатиме: одуль збору та аналізу мережевого трафіку. Модуль виявлення аномалій, що можуть свідчити про DDoS-атаку, на основі обраних методів (наприклад, статистичного аналізу, базових правил). Інтерфейс користувача для відображення даних моніторингу та результатів аналізу;

– провести тестування розробленого програмного забезпечення на змодельованому або реальному мережевому трафіку з метою оцінки його ефективності у виявленні DDoS-атак.

# 2 СПЕЦІАЛЬНИЙ РОЗДІЛ

## 2.1 Технічні вимоги до програмного забезпечення для моніторингу та виявлення DDoS-атак у мережевому трафіку комп’ютерних систем

### 2.1.1 Найменування і призначення програмного додатку

Призначенням програмного забезпечення "DDoS-Guard Monitor" є забезпечення можливості моніторингу мережевого трафіку комп’ютерних систем у реальному часі з метою виявлення аномалій, які можуть свідчити про здійснення розподіленої атаки типу "відмова в обслуговуванні" (DDoS). Програмне забезпечення призначене для:

– отримання інформації про мережеві пакети, що проходять через контрольовану систему або мережевий сегмент;

– обробка зібраних даних для виявлення патернів та характеристик, що відрізняються від нормального трафіку;

– ідентифікація відхилень від встановлених базових ліній поведінки мережі або відомих сигнатур атак;

– представлення інформації про мережевий трафік та результатів аналізу у зручному для користувача форматі (наприклад, таблиці, графіки);

– інформування користувача про виявлення ознак, що можуть свідчити про DDoS-атаку.

Програмне забезпечення "DDoS-Guard Monitor" може бути використане системними адміністраторами, фахівцями з інформаційної безпеки та організаціями, які прагнуть підвищити рівень захисту своїх інформаційних систем від DDoS-загроз шляхом своєчасного виявлення та реагування на атаки.

### 2.1.2 Вимоги до структури і функціонування програмного додатку

Програмне забезпечення "DDoS-Guard Monitor" розробляється з урахуванням модульної архітектури, що забезпечить його гнучкість, масштабованість та полегшить подальший розвиток. Ключовими структурними елементами

програмного забезпечення є модуль збору трафіку, модуль аналізу трафіку, модуль виявлення DDoS-атак, модуль візуалізації та модуль сповіщень (рис.2.1). Модуль збору трафіку відповідатиме за захоплення мережевих пакетів з визначених джерел, включаючи мережеві інтерфейси та файли дампів трафіку, з можливістю застосування фільтрів за основними параметрами мережі та ефективної обробки великих обсягів даних. Модуль аналізу трафіку реалізовуватиме різноманітні алгоритми для виявлення аномалій, підтримуватиме конфігурацію параметрів аналізу та можливість розширення набору методів у майбутньому. На основі результатів аналізу, модуль виявлення DDoS-атак ідентифікуватиме ознаки підозрілої активності, застосовуючи механізми кореляції та конфігуровані правила, а також генеруватиме відповідні події та сповіщення. Для зручного представлення інформації передбачено модуль візуалізації з графічним інтерфейсом, що відображатиме ключові метрики трафіку та результати аналізу з можливостями фільтрації та сортування. Нарешті, модуль сповіщень забезпечить інформування користувача про виявлені підозрілі події різними способами з можливістю налаштування параметрів сповіщень.

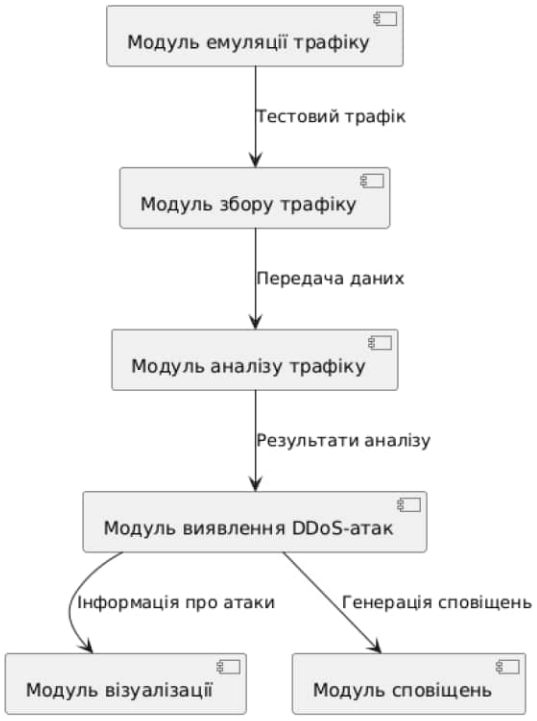



Рисунок 2.1 – Структурна схема елементів програмного забезпечення

Щодо функціонування, програмне забезпечення "DDoS-Guard Monitor" повинно працювати в режимі, близькому до реального часу, для своєчасного виявлення атак, ефективно використовуючи системні ресурси при обробці значних обсягів трафіку. Важливими вимогами є висока точність виявлення з мінімізацією хибних спрацьовувань, гнучкість у налаштуванні параметрів збору, аналізу та виявлення відповідно до потреб користувача. Архітектура програмного забезпечення повинна бути розширюваною, дозволяючи додавати нові функціональні можливості без значних змін. Зручність використання забезпечуватиметься інтуїтивно зрозумілим інтерфейсом, а стабільність та надійність функціонування є ключовими вимогами для тривалої роботи програмного забезпечення. Дотримання цих вимог до структури та функціонування є критично важливим для створення ефективного інструменту моніторингу та виявлення DDoS-атак.

### 2.1.3 Вимоги до до функцій, які виконує програмне забезпечення

Програмне забезпечення "DDoS-Guard Monitor" повинно забезпечувати наступний набір конкретних функцій для ефективного моніторингу та виявлення DDoS-атак.

Функції збору мережевого трафіку:

– програмне забезпечення повинно мати можливість безперервно прослуховувати визначений мережевий інтерфейс (наприклад, eth0, ens33) та захоплювати всі або фільтровані мережеві пакети, що проходять через нього. Приклад – користувач може обрати для моніторингу мережевий інтерфейс, підключений до зовнішнього каналу зв'язку сервера, щоб аналізувати вхідний трафік на предмет атак;

– програмне забезпечення повинно надавати можливість імпорту даних мережевого трафіку з файлів у форматі PCAP для подальшого аналізу (наприклад, для ретроспективного аналізу інцидентів або тестування алгоритмів виявлення). Приклад – адміністратор може завантажити файл PCAP, отриманий під час

попередньої підозрілої активності, щоб дослідити її характеристики за допомогою "DDoS-Guard Monitor";

– програмне забезпечення повинно дозволяти користувачеві визначати правила фільтрації трафіку на основі різних критеріїв, таких як IP-адреси джерела та призначення, порти TCP/UDP, протоколи (TCP, UDP, ICMP), для обмеження обсягу аналізованих даних та фокусування на потенційно цікавому трафіку. Приклад – користувач може налаштувати фільтр для аналізу лише трафіку, що надходить на веб-сервер (TCP порт 80 та 443) з певного діапазону IP-адрес, які раніше були помічені в підозрілій активності.

Функції аналізу мережевого трафіку:

– програмне забезпечення повинно відстежувати кількість пакетів, що надходять від кожної IP-адреси джерела за певний часовий інтервал, та сигналізувати про перевищення встановлених порогів. Приклад – якщо кількість SYN-пакетів від однієї IP-адреси перевищує 1000 за секунду, це може бути розцінено як ознака SYN-флуд атаки;

– програмне забезпечення повинно вимірювати швидкість передачі даних (в бітах або байтах за секунду) для кожної IP-адреси джерела та призначення, а також загальну швидкість трафіку, та виявляти різкі аномальні збільшення. Приклад – раптове зростання вхідного трафіку на веб-сервер у 10 разів може свідчити про об'ємну DDoS-атаку;

– програмне забезпечення повинно відстежувати кількість активних TCP/UDP з'єднань з кожної IP-адреси джерела до цільового сервера та виявляти аномально велику кількість нових або незавершених з'єднань. Приклад – велика кількість SYN-запитів без подальшого встановлення TCP-з'єднання може бути ознакою SYN-флуд атаки;

– програмне забезпечення повинно мати можливість аналізувати вміст заголовків мережевих пакетів на наявність специфічних ознак, характерних для певних типів атак (наприклад, встановлення нетипових комбінацій TCP-флагів, фрагментація IP-пакетів). Приклад – виявлення великої кількості ICMP-пакетів з однаковим ідентифікатором та невеликим розміром може бути ознакою ICMP-флуд

атаки.

Функції виявлення DDoS-атак:

– програмне забезпечення повинно дозволяти користувачеві встановлювати порогові значення для різних метрик трафіку (частота пакетів, швидкість передачі, кількість з'єднань) і генерувати сповіщення при їх перевищенні. Приклад – встановлення порогу в 500 нових TCP-з'єднань за секунду з однієї IP-адреси може ініціювати сповіщення про підозрілу активність;

– програмне забезпечення повинно підтримувати визначення правил виявлення DDoS-атак, що комбінують кілька умов на основі різних характеристик трафіку. Приклад – правило може визначати DDoS-атаку, якщо протягом 5 секунд кількість SYN-пакетів від однієї IP-адреси перевищує 300, а кількість встановлених з'єднань залишається низькою;

– програмне забезпечення повинно мати можливість аналізувати послідовність виявлених аномалій та корелювати їх для більш точного визначення DDoS-атак (наприклад, якщо одночасно фіксується різке зростання трафіку та велика кількість SYN-запитів).

Функції візуалізації:

– відображення графіків зміни загального обсягу трафіку (вхідного та вихідного) у часі;

– відображення графіків інтенсивності трафіку для окремих IP-адрес джерела та призначення;

– відображення таблиці активних TCP/UDP з'єднань з інформацією про IP-адреси, порти та стан з'єднання.

– візуальне виділення на графіках або в таблицях періодів часу або IP-адрес, для яких були зафіксовані аномальні значення метрик.

### 2.1.4 Вимоги до програмного забезпечення

Для забезпечення ефективного процесу розробки, зручного розгортання та подальшої стабільної роботи програмного забезпечення "DDoS-Guard Monitor",

призначеного для системних адміністраторів з метою моніторингу та виявлення DDoS-атак, висуваються наступні вимоги до використовуваного програмного забезпечення. На етапі розробки ключовим інструментом для серверної частини (Python) є інтегроване середовище розробки PyCharm, що забезпечує необхідні засоби для написання, налагодження та тестування коду, а для розробки веб-інтерфейсу (JavaScript) рекомендується Visual Studio Code завдяки його підтримці сучасних веб-технологій. Обов'язковим є використання системи контролю версій Git для управління кодом та спільної роботи над проектом. Для взаємодії з базами даних, які можуть використовуватись для зберігання конфігурації та логів, необхідний універсальний клієнт DBeaver, що підтримує різні СУБД.

На етапі розгортання критично важливим є використання платформи контейнеризації Docker. Docker дозволить упакувати всі компоненти "DDoS-Guard Monitor" разом з їхніми залежностями в ізольовані контейнери, що значно спростить процес встановлення та забезпечить консистентність роботи на різних серверах. Для доступу до веб-інтерфейсу моніторингу знадобиться будь-який сучасний веб-браузер, такий як Google Chrome або Mozilla Firefox. Крім того, для адміністрування сервера, на якому буде розгорнуто "DDoS-Guard Monitor", будуть потрібні стандартні інструменти командного рядка операційної системи, такі як ssh для віддаленого доступу та docker або docker-compose для керування контейнерами. Виконання цих вимог до програмного забезпечення забезпечить необхідну інфраструктуру для успішної розробки, зручного розгортання та ефективного використання "DDoS-Guard Monitor" системними адміністраторами.

## 2.2 Програмування додатку для моніторингу та виявлення DDoS-атак у мережевому трафіку КС

### 2.2.1 Призначення програмного додатку

Програмне забезпечення "DDoS-Guard Monitor" розробляється для інтеграції у функціональний блок моніторингу та виявлення загроз безпеки досліджуваних комп'ютерних систем та їхнього мережевого трафіку. Його ключове призначення полягає у забезпеченні безперервного, автоматизованого спостереження за

мережевою активністю з метою своєчасної ідентифікації аномальних патернів, які можуть свідчити про спробу здійснення розподіленої атаки типу "відмова в обслуговуванні" (DDoS).

Для досягнення цієї мети "DDoS-Guard Monitor" виконує ряд важливих функцій у рамках блоку моніторингу та виявлення загроз. По-перше, забезпечується моніторинг мережевого трафіку, що включає постійний збір детальних даних про мережеві пакети, які передаються в межах досліджуваної мережевої інфраструктури або на окремих контрольованих вузлах. По-друге, здійснюється аналіз трафіку на предмет аномалій, в рамках якого відбувається автоматизована обробка зібраних даних з метою виявлення відхилень від усталених нормальних патернів мережевої комунікації із застосуванням статистичних методів та попередньо визначених правил. Третьою ключовою функцією є виявлення ознак DDoS-атак, що передбачає ідентифікацію специфічних характеристик мережевого трафіку, які корелюють з відомими типами розподілених атак, такими як флуд-атаки, протокольні атаки або атаки на рівні прикладних протоколів. Для забезпечення зручності інтерпретації отриманих даних передбачена візуалізація даних моніторингу та виявлених загроз, що реалізується через інтуїтивно зрозумілий графічний інтерфейс, який відображає ключові показники мережевого трафіку та результати аналізу, полегшуючи розуміння поточної ситуації з безпекою мережевих ресурсів. Нарешті, важливою функцією є сповіщення про потенційні DDoS-атаки, що полягає в оперативному інформуванні системних адміністраторів або відповідальних за безпеку осіб про виявлення підозрілої мережевої активності, яка може становити безпосередню загрозу доступності критично важливих мережевих сервісів та ресурсів.

Алгоритм роботи програмного додатку "DDoS-Guard Monitor" представлено на рис.2.2 та базується на безперервному циклі збору, аналізу та реагування на мережевий трафік з метою виявлення ознак DDoS-атак. При запуску програмний додаток ініціалізує свої модулі та завантажує початкові налаштування, включаючи визначені мережеві інтерфейси для моніторингу, параметри фільтрації трафіку, порогові значення для статистичного аналізу та правила виявлення DDoS-атак. Користувач має можливість змінювати ці налаштування через веб-інтерфейс.

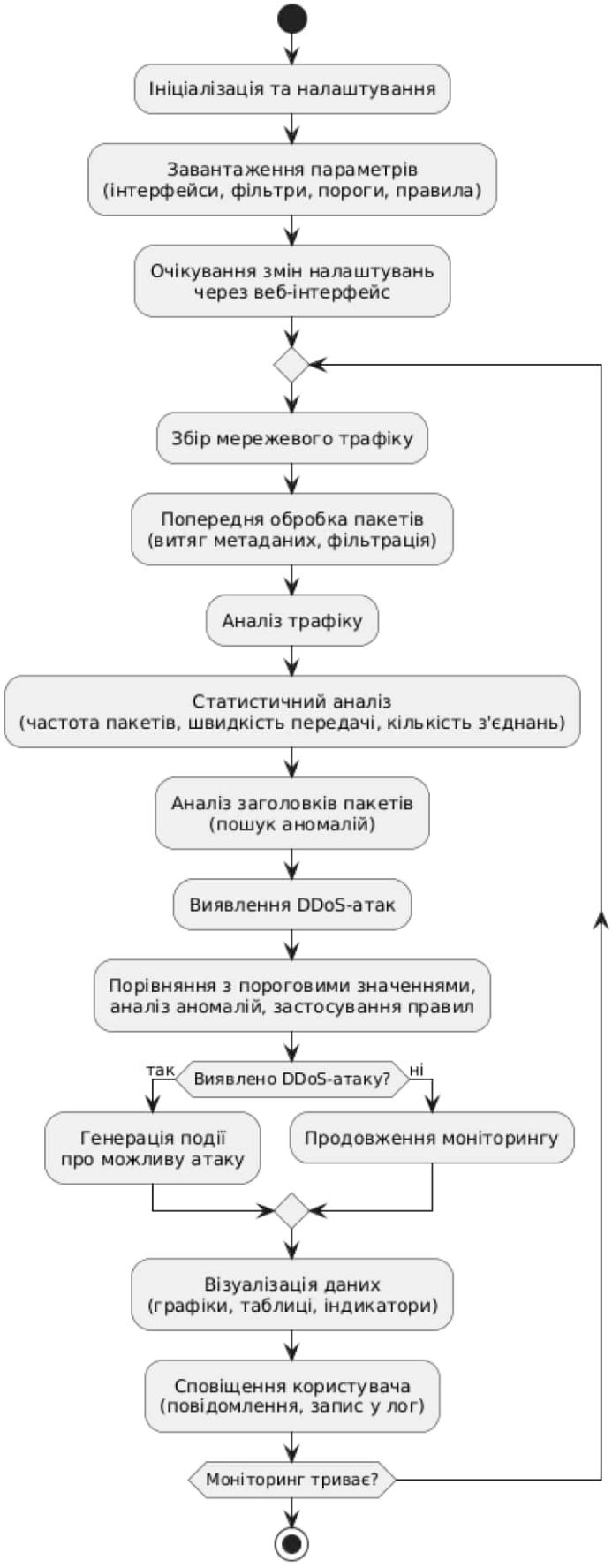



Рисунок 2.2 – Алгоритм роботи програмного додатку

Активується модуль збору трафіку, який починає прослуховувати визначені мережеві інтерфейси або обробляти завантажений файл PCAP. Захоплені пакети передаються на етап попередньої обробки, де з них вилучаються необхідні метадані (IP-адреси джерела та призначення, порти, протоколи, розміри пакетів, TCP-флаги тощо). За потреби, застосовуються налаштовані фільтри для виключення нерелевантного трафіку.

Оброблений трафік надходить до модуля аналізу, де застосовуються різні методи для виявлення аномалій. Це включає статистичний аналіз (наприклад, розрахунок частоти пакетів, швидкості передачі даних, кількості нових та активних з'єднань для кожної IP-адреси), а також аналіз вмісту заголовків пакетів на наявність підозрілих комбінацій або значень. Результати цього аналізу передаються до модуля виявлення DDoS-атак.

Модуль виявлення DDoS-атак використовує отримані від модуля аналізу дані та застосовує налаштовані порогові значення та правила для ідентифікації ознак DDoS-атак. Це може включати порівняння поточних значень метрик з встановленими порогами, аналіз комбінацій різних аномалій та застосування правил, що описують характерні патерни відомих типів атак. У разі виявлення підозрілої активності, генерується подія про можливу DDoS-атаку.

Інформація про поточний трафік та виявлені аномалії передається до модуля візуалізації для відображення на веб-інтерфейсі у вигляді графіків, таблиць та індикаторів. Одночасно модуль сповіщень інформує користувача про виявлені підозрілі події шляхом відображення повідомлень в інтерфейсі та/або запису відповідних записів у лог-файл.

Після етапу візуалізації та сповіщення система повертається до етапу збору мережевого трафіку, забезпечуючи безперервний моніторинг. Користувач може в будь-який момент змінити налаштування системи через веб-інтерфейс, що вплине на подальші цикли аналізу та виявлення

### 2.2.2 Інтерфейс користувача

Програмний додаток для моніторингу та виявлення DDoS-атак "DDoS Емулятор і Виявлення" має інтуїтивно зрозумілий інтерфейс, виконаний у темних тонах (рис.2.3). У верхній частині головного вікна розміщено заголовок, що чітко вказує на призначення програми. Основна частина екрану містить вертикально розташований набір кнопок, кожна з яких відповідає за певний функціональний модуль програми. У нижній частині вікна знаходиться область, що наразі представлена порожнім графіком з позначеними осями, яка призначена для візуалізації даних, отриманих в процесі роботи програми.


Рисунок 2.3 – Інтерфейс програмного додатку

Першим елементом інтерфейсу є текстовий індикатор "Аналіз пакетів запущено", виконаний сірим кольором, який, ймовірно, відображає поточний стан програми або повідомляє про останню виконану дію, сигналізуючи про те, що

процес аналізу мережевих пакетів був активований.

Далі розташована червона кнопка "Запустити атаку", натискання на яку ініціює процес імітації DDoS-атаки. Після активації цієї функції користувачеві, ймовірно, буде запропоновано обрати конкретний тип атаки та налаштувати її параметри, такі як кількість запитів, інтенсивність їх надсилання та цільову IP-адресу з відповідним портом.

Наступною є синя кнопка "Виявити атаку", яка відповідає за запуск модуля, призначеного для аналізу мережевого трафіку з метою виявлення характерних ознак DDoS-атак. Ця функція може бути використана як після імітації атаки, так і при аналізі реальних даних мережевого трафіку.

Бірюзова кнопка "Трафік в реальному часі" активує режим відображення динаміки мережевого трафіку в реальному часі. Після її натискання в нижній графічній області екрану має відображатися зміна ключових характеристик трафіку, таких як кількість пакетів за секунду або швидкість передачі даних.

Зелена кнопка "Візуалізувати трафік" призначена для відображення вже зібраних або попередньо проаналізованих даних мережевого трафіку у вигляді різноманітних графіків та діаграм у нижній частині вікна. Користувач, ймовірно, матиме можливість вибору параметрів трафіку для візуалізації та типу графічного представлення.

Помаранчева кнопка "Збір трафіку" відкриває інтерфейс для детального дослідження окремих мережевих пакетів. Після її натискання користувач зможе переглядати вміст заголовків пакетів різних рівнів моделі OSI та їх корисне навантаження.

Завершує перелік кнопок елемент "Вийти", виконаний сірим кольором, який призначений для завершення роботи з програмним додатком.

Фрагмент програмного коду на рис.2.4, що супроводжує опис інтерфейсу, демонструє реалізацію основних функцій програми на мові Python з використанням бібліотек Tkinter (ttkbootstrap для стилізації) та Matplotlib для візуалізації. Код включає класи для графічного інтерфейсу (DdosEmulatorGUI) та логіки виявлення атак (DdosDetector). Функціональність програми охоплює запуск імітованої атаки,

виявлення атаки шляхом аналізу часових інтервалів між запитами та протоколів, відображення трафіку в реальному часі у вигляді графіка, візуалізацію мережевого трафіку з побудовою графа зв'язків, запуск окремого процесу для аналізу пакетів (хоча його конкретна реалізація не представлена в цьому фрагменті коду), а також логування виявлених атак у файл. Клас DdosDetector відповідає за запис часу та протоколу кожного запиту та аналіз трафіку на предмет аномалій, що можуть свідчити про DDoS-атаку, базуючись на середньому інтервалі між запитами та співвідношенні TCP/UDP та ICMP протоколів.

```
class DdosEmulatorGUI:
    def __init__(self, root):
        self.root = root
        self.root.title("DDoS Емулятор і Виявлення")
        self.root.geometry("900x650")
        self.style = tb.Style("superhero")

        self.ddos_detector = DdosDetector()
        self.traffic_data = []
        self.protocols = []  # Зберігаємо типи мережевих протоколів

        self.frame = ttk.Frame(root, padding=20)
        self.frame.pack(expand=True, fill='both')

        self.label = ttk.Label(self.frame, text="DDoS Емулятор", font=("Helvetica", 16))
        self.label.pack(pady=10)

        self.attack_button = ttk.Button(self.frame, text="Запустити атаку", command=self.start_attack,
                                        style="primary.TButton")
        self.attack_button.pack(pady=10)

        self.detect_button = ttk.Button(self.frame, text="Виявити атаку", command=self.detect_attack,
                                        style="danger.TButton")
        self.detect_button.pack(pady=10)

        self.real_time_button = ttk.Button(self.frame, text="Трафік в реальному часі",
                                           command=self.toggle_real_time_traffic, style="info.TButton")
        self.real_time_button.pack(pady=10)

        self.visualize_button = ttk.Button(self.frame, text="Візуалізувати трафік",
                                           command=self.visualize_network_traffic, style="success.TButton")
        self.visualize_button.pack(pady=10)

        self.packet_analysis_button = ttk.Button(self.frame, text="Аналіз пакетів", command=self.run_packet_analysis,
                                                 style="warning.TButton")
        self.packet_analysis_button.pack(pady=10)

        self.quit_button = ttk.Button(self.frame, text="Вийти", command=root.quit, style="secondary.TButton")
        self.quit_button.pack(pady=10)

        self.fig, self.ax = plt.subplots(figsize=(6, 3))
        self.canvas = FigureCanvasTkAgg(self.fig, master=self.frame)
        self.canvas.get_tk_widget().pack()

        self.is_updating = False  # Для перевірки, чи оновлюємо графік
        self.traffic_data = []  # Список для збереження часів запитів
        self.protocols = []  # Список для збереження протоколів

    def start_attack(self):
```

Рисунок 2.4 – Фрагмент програмного коду для створення інтерфейсу

### 2.2.3 Модуль емуляції DDoS-атак

Основною метою модуля "Запустити атаку" є імітація різних типів розподілених атак типу "відмова в обслуговуванні" (DDoS). Це дозволяє користувачам тестувати ефективність систем виявлення атак, аналізувати поведінку мережевих пристроїв під час атаки та глибше розуміти механізми дії різних видів DDoS. Модуль надає контрольоване середовище для генерації шкідливого трафіку, що імітує реальні атаки, без необхідності використання справжніх мережевих ресурсів для атаки.

Модуль "Запустити атаку" надає користувачеві ряд опцій для налаштування процесу імітації DDoS-атаки (рис.2.5). На даному етапі розробки, згідно з представленим програмним кодом, реалізована базова імітація атаки шляхом генерації випадкових запитів з протоколами TCP, UDP та ICMP.

```
def start_attack(self):
        self.traffic_data.clear()
        self.protocols.clear()
        self.label.config(text="Запускаємо атаку...")
        thread = threading.Thread(target=self.simulate_attack, daemon=True)
        thread.start()

    def simulate_attack(self):
        for _ in range(100):
            timestamp = time.time()
            protocol = random.choice(["TCP", "UDP", "ICMP"])  # Випадковий протокол
            self.ddos_detector.record_request(timestamp, protocol)
            self.traffic_data.append(timestamp)
            self.protocols.append(protocol)
            time.sleep(random.uniform(0.01, 0.05))  # дуже короткий інтервал для атаки
        self.label.config(text="Атака завершена!")
        self.update_graph()
```

Рисунок 2.5 – Фрагмент програмного коду для модуля емуляції DDoS-атак

Наведений фрагмент коду на рис.2.5 демонструє базову реалізацію функції "Запустити атаку". При натисканні на відповідну кнопку очищаються попередні дані трафіку та протоколів, змінюється текст мітки стану, і в окремому потоці запускається функція simulate_attack. Ця функція імітує атаку шляхом генерації 100 випадкових запитів з протоколами TCP, UDP або ICMP з дуже короткими випадковими інтервалами між ними, що імітує високу інтенсивність атаки. Для кожного запиту фіксується час та протокол. Після завершення циклу імітації оновлюється текст мітки стану та графік відображення трафіку. У майбутньому цей код буде розширено для реалізації вибору типу атаки та налаштування її параметрів

через відповідні елементи інтерфейсу користувача.

### 2.2.4 Модуль виявлення атак

Основною метою модуля "Виявити атаку" є аналіз мережевого трафіку для ідентифікації характерних ознак, що свідчать про здійснення розподіленої атаки типу "відмова в обслуговуванні" (DDoS). Цей модуль є ключовим компонентом програми, що дозволяє оцінити ефективність методів виявлення атак на згенерованому трафіку або на реальних мережевих даних. Його функціональність спрямована на автоматичне виявлення аномалій та патернів, які відрізняються від нормального мережевого обміну, та класифікацію їх як потенційні DDoS-атаки (рис.2.6).

Модуль "Виявити атаку" надає наступні можливості для аналізу трафіку та виявлення DDoS-атак. Модуль підтримує аналіз трафіку з різних джерел. По-перше, це трафік, згенерований вбудованим модулем емуляції атак ("Запустити атаку"). По-друге, передбачається можливість імпорту даних мережевого трафіку з зовнішніх файлів у форматі PCAP, що дозволяє аналізувати раніше записаний трафік або трафік, зібраний іншими інструментами. У поточному програмному коді аналізується трафік, згенерований модулем емуляції та збережений у внутрішніх структурах даних (self.traffic_data, self.protocols).

В основі модуля лежить реалізація різних алгоритмів виявлення аномалій та, можливо, сигнатурних правил, спрямованих на ідентифікацію DDoS-атак. У представленому коді реалізовано простий статистичний метод, що базується на аналізі середнього інтервалу між запитами та співвідношенні кількості TCP/UDP та ICMP протоколів. Якщо середній інтервал між запитами є дуже малим, а кількість TCP/UDP пакетів значно перевищує кількість ICMP пакетів, це розцінюється як ознака DDoS-атаки (TCP/UDP).

Для забезпечення гнучкості та адаптивності системи виявлення модуль повинен надавати користувачеві можливість налаштування ключових параметрів алгоритмів виявлення. У поточному коді жорстко задані порогове значення для

середнього інтервалу між запитами (менше 0.05 секунди) та умова порівняння кількості протоколів.

Після завершення аналізу трафіку модуль повинен надавати користувачеві інформацію про результати виявлення. У поточному коді при виявленні атаки змінюється текст мітки стану та виводиться попереджувальне вікно з типом виявленої атаки. Також передбачено логування інформації про виявлену атаку у файл.

```
def detect_attack(self):
        self.label.config(text="Аналізуємо...")
        is_ddos, attack_type = self.ddos_detector.analyze_traffic()
        if is_ddos:
            self.label.config(text=f"Виявлена DDoS-атака ({attack_type})!", foreground="red")
            messagebox.showwarning("Попередження", f"Виявлено DDoS-атаку ({attack_type})!")
            self.log_attack(attack_type)  # Логування атаки
        else:
            self.label.config(text="Атака не виявлена")
            messagebox.showinfo("Результат", "DDoS-атака не виявлена")
        self.update_graph()

class DdosDetector:
    def __init__(self):
        self.timestamps = []
        self.protocols = []

    def record_request(self, timestamp, protocol):
        self.timestamps.append(timestamp)
        self.protocols.append(protocol)

    def analyze_traffic(self):
        if len(self.timestamps) < 10:
            return False, ""  # Якщо немає достатньо даних, атака не виявлена

        intervals = np.diff(self.timestamps)
        avg_interval = np.mean(intervals) if len(intervals) > 0 else float('inf')

        # Підрахунок протоколів
        tcp_udp_count = sum(1 for p in self.protocols if p in ['TCP', 'UDP'])
        icmp_count = self.protocols.count('ICMP')

        # Умови для атаки
        if avg_interval < 0.05 and tcp_udp_count > icmp_count:
            return True, "DDoS-атака (TCP/UDP)"
        else:
            return False, "Немає атаки (ICMP)"
```

Рисунок 2.6 – Фрагмент програмного коду для модуля виявлення атак

Наведений фрагмент коду на рис.2.6 демонструє поточну реалізацію модуля виявлення атак. При натисканні кнопки "Виявити атаку" змінюється текст мітки стану та викликається метод analyze_traffic класу DdosDetector. Цей метод аналізує збережені часові мітки запитів та протоколи для виявлення ознак DDoS-атаки на основі простого статистичного аналізу середнього інтервалу між запитами та співвідношення протоколів. Залежно від результату аналізу, оновлюється текст мітки стану, виводиться відповідне повідомлення та, у разі виявлення атаки,

здійснюється її логування. Також викликається функція оновлення графіка для відображення трафіку. У майбутньому цей модуль буде значно розширено для реалізації більш складних алгоритмів виявлення та надання ширших можливостей для налаштування параметрів аналізу.

### 2.2.5 Модуль моніторингу трафіку в реальному часі

Основною метою модуля "Трафік в реальному часі" є забезпечення безперервного відображення динаміки мережевого трафіку в режимі, наближеному до реального часу. Це дозволяє користувачеві спостерігати за поточною мережевою активністю, виявляти аномальні сплески трафіку або інші нехарактерні зміни, які можуть бути ознаками проблем у мережі або початку DDoS-атаки (рис.2.7). Модуль надає візуальне представлення ключових метрик мережевого трафіку, що полегшує його розуміння та оперативний контроль. Цей модуль забезпечує безперервне отримання даних безпосередньо з обраних мережевих інтерфейсів комп'ютерної системи, на якій запущено додаток. Це передбачає використання бібліотек для роботи з мережевими пакетами (наприклад, pурcap, scapy). При активації режиму "Трафік в реальному часі" запускається окремий потік (update_traffic), який з певною періодичністю додає нові випадкові дані до внутрішніх структур (self.traffic_data, self.protocols) та оновлює графік. У майбутньому цей модуль буде інтегровано з бібліотеками захоплення мережевого трафіку для отримання реальних даних з мережевих інтерфейсів [11,12].

Модуль відповідає за відображення важливих характеристик мережевого трафіку у вигляді графіків. У поточному коді реалізовано відображення часової серії надходження запитів, де вісь X представляє час, а вісь Y – порядковий номер запиту. Кожен протокол (TCP, UDP, ICMP) відображається окремим кольором.

Для полегшення аналізу великих обсягів трафіку модуль повинен надавати користувачеві можливість фільтрації відображуваних даних за різними критеріями.

Однією з важливих функцій модуля є візуальне виділення аномальних змін трафіку, які можуть свідчити про початок DDoS-атаки. Реалізовано базове

автоматичне виявлення атаки під час оновлення трафіку та зміна кольору мітки стану при її виявленні.

```
def toggle_real_time_traffic(self):
        if self.is_updating:
            self.is_updating = False
            self.label.config(text="Трафік в реальному часі зупинено.")
            self.real_time_button.config(text="Запустити трафік в реальному часі")
        else:
            self.label.config(text="Трафік в реальному часі...")
            self.is_updating = True
            self.traffic_data.clear()  # Очищаємо попередні дані
            self.protocols.clear()
            thread = threading.Thread(target=self.update_traffic, daemon=True)
            thread.start()
            self.real_time_button.config(text="Зупинити трафік")

    def update_traffic(self):
        while self.is_updating:
            timestamp = time.time()
            protocol = random.choice(["TCP", "UDP", "ICMP"])
            self.ddos_detector.record_request(timestamp, protocol)
            self.traffic_data.append(timestamp)
            self.protocols.append(protocol)

            # Автоматичне виявлення атаки під час оновлення трафіку
            is_ddos, attack_type = self.ddos_detector.analyze_traffic()

            # Визначення інтервалу між запитами для нормального трафіку
            if len(self.traffic_data) > 1:
                intervals = [self.traffic_data[i] - self.traffic_data[i-1] for i in range(1, len(self.traffic_data))]
                avg_interval = sum(intervals) / len(intervals)
            else:
                avg_interval = 0

            if is_ddos:
                self.label.config(text=f"Виявлена DDoS-атака ({attack_type})!", foreground="red")
                messagebox.showwarning("Попередження", f"Виявлено DDoS-атаку ({attack_type})!")
                self.log_attack(attack_type)  # Логування атаки
            else:
                if avg_interval < 0.1:
                    self.label.config(text=f"Нормальний трафік: інтервал запитів {avg_interval:.2f} секунд, високий рівень за
                else:
                    self.label.config(text=f"Нормальний трафік: інтервал запитів {avg_interval:.2f} секунд.", foreground="gre

            self.update_graph()
            time.sleep(1)  # Оновлення кожну секунду

    def update_graph(self):
        self.ax.clear()
        if self.traffic_data:
```

Рисунок 2.7 – Фрагмент програмного коду для модуля моніторингу трафіку в реальному часі

### 2.2.6 Модуль збору трафіку

Модуль збору трафіку відповідає за отримання даних, що представляють мережеві пакети, для подальшого аналізу та виявлення потенційних загроз, зокрема DDoS-атак. У контексті наданого програмного коду, який імітує роботу URH (Universal Radio Hacker), цей модуль виконує функцію генерації випадкових даних, що моделюють характеристики мережевих пакетів (рис.2.8). У повноцінній реалізації системи моніторингу DDoS, цей модуль мав би забезпечувати захоплення реального мережевого трафіку з обраних інтерфейсів.

Модуль збору трафіку в програмному забезпеченні "DDoS Емулятор і Виявлення" вже реалізовано з використанням бібліотеки scapy. Він забезпечує безперервне захоплення мережевих пакетів з обраного мережевого інтерфейсу. Кожен захоплений пакет піддається розбору, що дозволяє отримати доступ до його заголовків різних рівнів (Ethernet, IP, TCP/UDP, ICMP тощо) та основних полів, таких як час отримання, IP-адреси джерела та призначення, порти та протоколи.

Користувач має можливість інтерактивно обирати мережевий інтерфейс своєї системи, трафік якого він бажає відстежувати.

Після вибору інтерфейсу модуль переходить у режим безперервного прослуховування, використовуючи функціональність sniff бібліотеки scapy. Це забезпечує отримання копій всіх мережевих пакетів, що проходять через вказаний інтерфейс, в режимі реального часу.

Однією з ключових переваг використання scapy є її здатність автоматично розпізнавати та розбирати заголовки різних рівнів мережевої моделі OSI. Для кожного захопленого пакету scapy створює відповідні об'єкти (наприклад, Ether, IP, TCP, UDP, ICMP), що надає зручний структурований доступ до полів протоколів, таких як IP-адреси джерела та призначення, порти, типи протоколів, TCP-флаги тощо.

Модуль підтримує можливість застосування фільтрів трафіку на етапі захоплення. Користувач може визначити фільтри на основі синтаксису, що використовується утилітою tcpdump (який також підтримується scapy), для відбору лише певних типів пакетів, що відповідають заданим критеріям (наприклад, трафік з певної IP-адреси, трафік на певний порт, пакети певного протоколу). Це дозволяє зменшити навантаження на систему та зосередитися на аналізі потенційно шкідливого трафіку.

Захоплені та розібрані об'єкти пакетів scapy у реальному часі передаються до наступних модулів програми, зокрема до модуля аналізу трафіку та модуля виявлення атак. Ці модулі використовують структуровані дані пакетів для виявлення аномалій та характерних ознак DDoS-атак [13,14].

```
from scapy.all import sniff, IP, TCP, UDP, ICMP
import threading
import time
from collections import deque

class TrafficCollector:
    def __init__(self, buffer_size=1000):
        self.is_capturing = False
        self.capture_thread = None
        self.interface = None
        self.packet_buffer = deque(maxlen=buffer_size)
        self.filter = ""

    def start_capture(self, interface, filter=""):
        self.interface = interface
        self.filter = filter
        self.is_capturing = True
        self.capture_thread = threading.Thread(target=self._capture_packets, daemon=True)
        self.capture_thread.start()
        print(f"Початок захоплення трафіку на інтерфейсі '{self.interface}' з фільтром '{self.filter}'...")

    def stop_capture(self):
        self.is_capturing = False
        if self.capture_thread and self.capture_thread.is_alive():
            print("Зупинка захоплення трафіку...")
            # Немає надійного способу примусово зупинити sniff в scapy,
            # тому покладаємося на умову self.is_capturing в callback.
            self.capture_thread.join(timeout=5)
            if self.capture_thread.is_alive():
                print("Потік захоплення не зупинився вчасно.")
            else:
                print("Захоплення трафіку зупинено.")

    def get_next_packet(self, timeout=None):
        if self.packet_buffer:
            return self.packet_buffer.popleft()
        elif timeout is not None:
            time.sleep(timeout)
            if self.packet_buffer:
                return self.packet_buffer.popleft()
        return None

    def _capture_packets(self):
        def packet_callback(packet):
            if self.is_capturing:
                self.packet_buffer.append(packet)
                # Тут можна було б одразу передавати пакет на обробку,
                # але для прикладу ми додаємо його до буфера.
```

Рисунок 2.8 – Фрагмент програмного коду для модуля збору трафіку

### 2.2.7 Модуль візуалізації трафіку

Основною метою модуля "Візуалізувати трафік" є надання користувачеві наочного графічного представлення зібраних та проаналізованих даних мережевого трафіку. Візуалізація є важливим інструментом для розуміння великих обсягів даних, виявлення закономірностей, аномалій та оцінки ефективності роботи системи моніторингу DDoS. Модуль має забезпечувати гнучкість у виборі параметрів для відображення та типів графіків, а також надавати можливості для налаштування зовнішнього вигляду візуалізацій та їх збереження для подальшого аналізу або звітності.

Користувач має можливість обирати, які саме характеристики мережевого

трафіку будуть відображені графічно:

– відображення обсягу трафіку, що генерується або спрямовується до певних IP-адрес (джерел або призначень). Це може бути корисно для ідентифікації найбільш активних вузлів або потенційних джерел/цілей атаки;

– відображення обсягу трафіку, що використовує певні порти TCP або UDP. Аналіз трафіку за портами може допомогти виявити атаки, спрямовані на конкретні сервіси;

– відображення частки різних мережевих протоколів (TCP, UDP, ICMP, HTTP, DNS тощо) у загальному обсязі трафіку. Різкі зміни у розподілі протоколів можуть бути ознакою атаки;

– відображення зміни обсягу трафіку (кількість пакетів або байт за секунду) протягом певного періоду часу. Графіки часових рядів можуть допомогти виявити аномальні сплески трафіку, характерні для DDoS-атак;

– відображення динаміки встановлення нових мережевих з'єднань, особливо TCP-з'єднань, що може бути індикатором SYN-флуду;

– залежно від реалізованих модулів аналізу, можуть бути доступні для візуалізації й інші метрики, такі як кількість відкинутих пакетів, затримки, тощо.

Модуль підтримує різні типи графічного представлення даних для забезпечення найбільш наочного відображення обраних параметрів. Після запуску атаки або в процесі її виконання користувачеві може бути надана можливість візуалізації характеристик згенерованого атакуючого трафіку в реальному часі або після її завершення. У поточному інтерфейсі нижня графічна область використовується для відображення загального трафіку, включаючи імітовану атаку.

## 2.3 Результати програмної реалізації додатку для моніторингу та виявлення DDoS-атак

Після успішної програмної реалізації додаток "DDoS Емулятор і Виявлення" надає користувачеві інтуїтивно зрозумілий інтерфейс для виконання ключових завдань, пов'язаних з моніторингом та виявленням DDoS-атак. Запуск програми

призводить до відображення головного вікна з набором функціональних кнопок та областю для візуалізації даних.

Користувач запускає виконуваний файл програми (наприклад, main.py при використанні Python). Відкривається головне вікно програми з заголовком "DDoS Емулятор і Виявлення". У центральній частині вікна відображаються кнопки для доступу до основних функцій: "Запустити атаку", "Виявити атаку", "Трафік в реальному часі", "Візуалізувати трафік", "Аналіз пакетів", "Вийти". Нижня частина вікна містить порожню область графіка, готову для відображення даних.

Користувач натискає кнопку "Трафік в реальному часі". Програма автоматично налаштована на мережевий інтерфейс для моніторингу. В нижній області вікна починає відображатися графік динаміки мережевого трафіку в реальному часі (рис.2.9). На графіку можуть відображатися такі параметри, як кількість пакетів за секунду, швидкість передачі даних, розподіл трафіку за протоколами тощо.

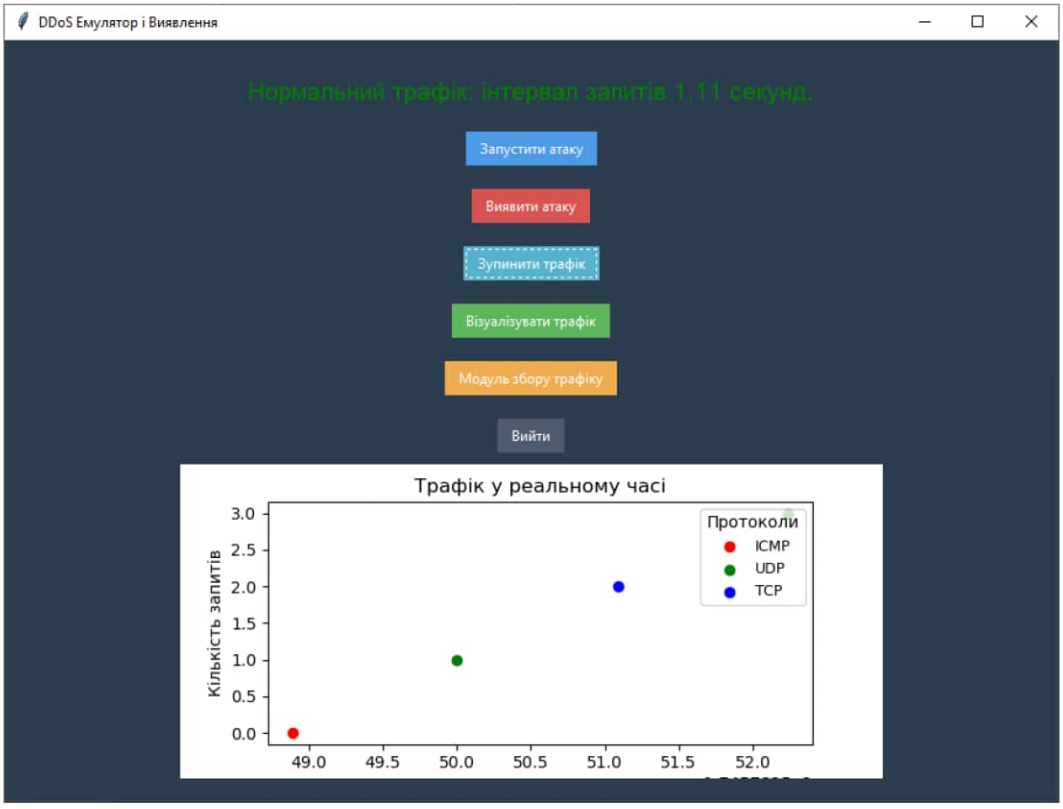



Рисунок 2.9 – Результат моніторингу трафіку в реальному часі

Користувач може спостерігати за змінами трафіку, виявляючи аномальні сплески або інші підозрілі патерни. При виявленні програмою потенційної атаки, в інтерфейсі може з'являтися відповідне сповіщення (текстове повідомлення). Користувач може натиснути кнопку "Зупинити трафік" для припинення моніторингу в реальному часі.

Графік на рис.2.9 показує надходження мережевих запитів у часі. Кожна точка представляє окремий запит, а її колір вказує на протокол цього запиту. Ми бачимо, що за останній відображений проміжок часу було зафіксовано кілька TCP-запитів та один ICMP-запит. Програма відображає динаміку надходження мережевих запитів, розрізняючи їх за протоколами ICMP та TCP. Зелений напис вгорі вказує на те, що на даний момент трафік розцінюється як нормальний, з середнім інтервалом між запитами близько 1.11 секунди. Користувач може спостерігати за цим графіком для виявлення будь-яких різких змін в інтенсивності трафіку або аномального співвідношення протоколів, що може бути ознакою DDoS-атаки. Кнопка "Зупинити трафік" надає можливість припинити моніторинг у реальному часі.

Імітація DDoS-атаки (рис.2.10). Користувач натискає кнопку "Запустити атаку". Програма генерує імітований атакуючий трафік. Динаміка цього трафіку може відображатися в області графіка в реальному часі. Після завершення імітації програма повідомляє про це користувача.

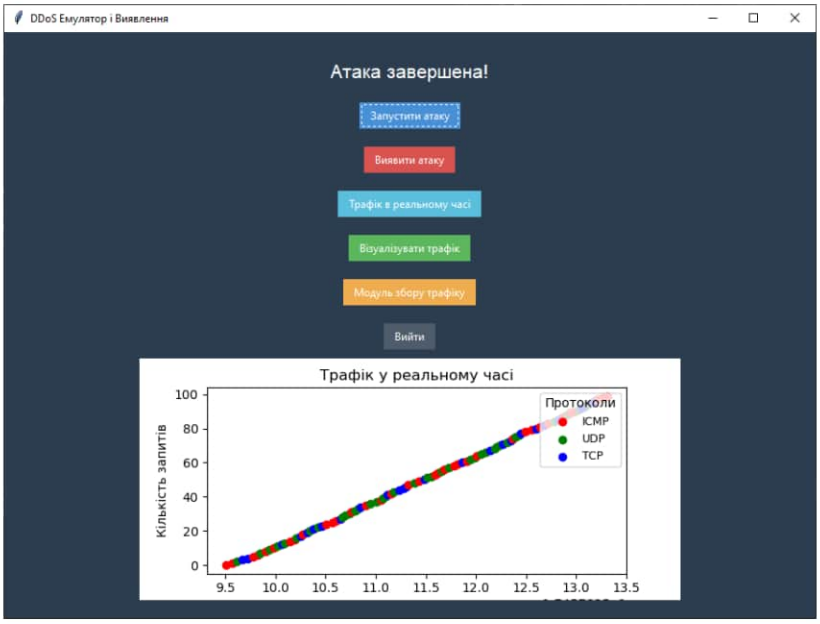



Рисунок 2.10 – Результат імітації DDoS-атаки

Після імітації атаки або під час моніторингу реального трафіку користувач натискає кнопку "Виявити атаку". Програма аналізує зібрані дані трафіку на предмет наявності ознак DDoS-атаки, використовуючи реалізовані алгоритми виявлення (наприклад, аналіз інтенсивності трафіку, частоти з'єднань, розподілу протоколів). Результати аналізу відображаються в інтерфейсі. Якщо атаку виявлено, програма сповіщає користувача про це (наприклад, через зміну тексту мітки, виведення діалогового вікна, відображення інформації на графіку). Може бути вказано тип виявленої атаки та час її виявлення.

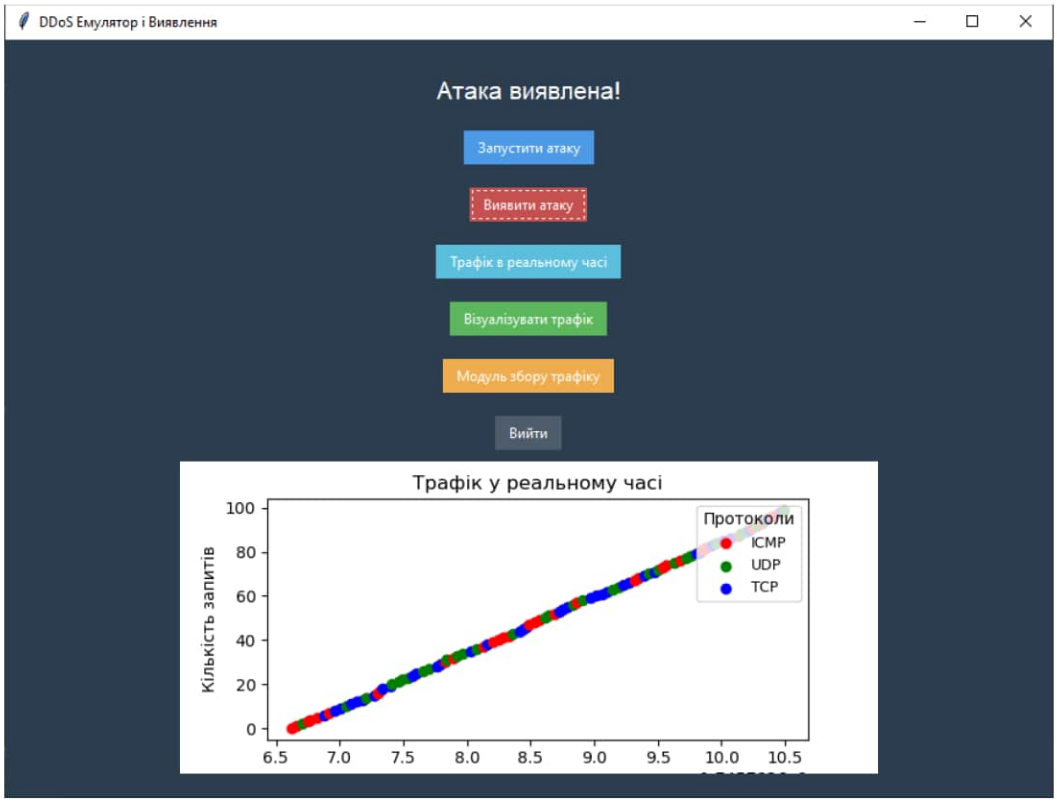



Рисунок 2.11 – Результат виявлення атаки

Модуль збору трафіку. Основний елемент інтерфейсу модуля представлений у вигляді таблиці, кожен рядок якої відображає окремий симульований пакет (рис.2.12). Таблиця структурована за стовпцями, що відповідають різним атрибутам мережевого пакету. Серед відображуваних характеристик можна побачити преамбулу (Preamble), байт синхронізації (Sync), довжину пакету в байтах (Length), тип пакету (Type), адреси призначення (DST Address) та джерела (SRC Address),

порядковий номер (Seq Num), а також штучно заданий рівень загрози (Threat Level). Значення у більшості стовпців представлені у шістнадцятковому форматі, що є типовим для відображення низькорівневих мережевих даних. Стовпець "Threat Level" містить текстові мітки, що вказують на потенційний рівень небезпеки, асоційований з кожним імітованим пакетом ("Threat", "Suspicious", "Normal").

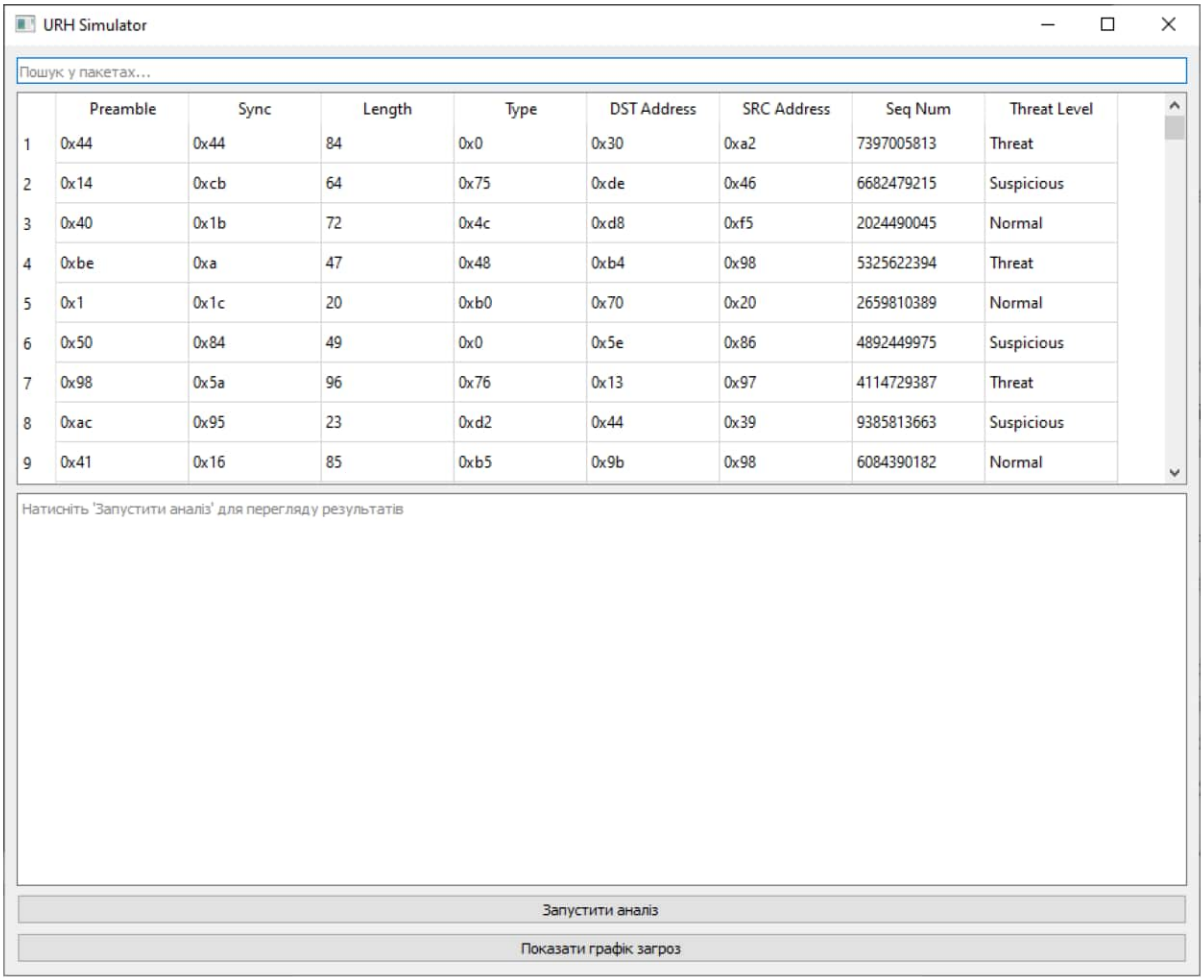
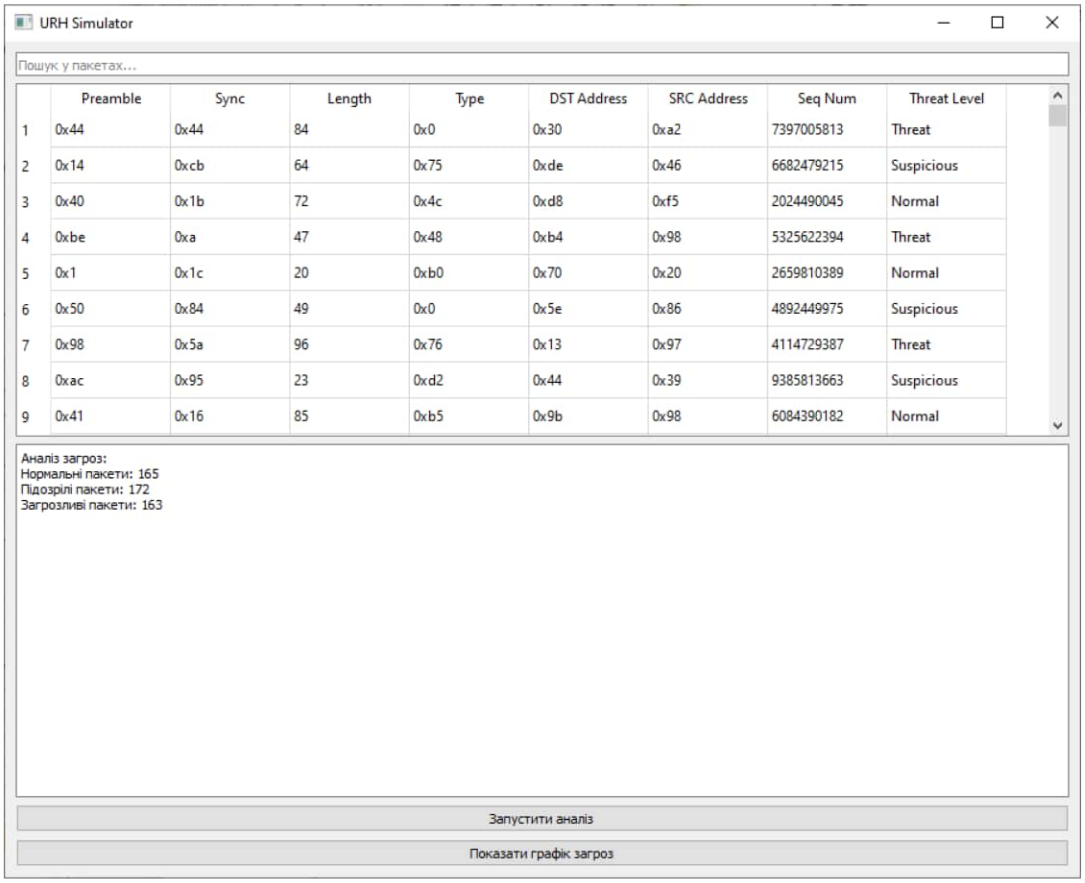



Рисунок 2.12 – Результат запуска модуля збору трафіку

Після натискання кнопки "Запустити аналіз" симулятор проаналізував відображені в таблиці симульовані мережеві пакети та підрахував кількість пакетів кожного рівня загрози (рис.2.13). Результати цього аналізу представлені у блоці "Аналіз загроз". Ми бачимо, що серед проаналізованих пакетів найбільшу кількість становлять "Підозрілі пакети" (172), за ними йдуть "Нормальні пакети" (165), а найменша кількість - "Загрозливі пакети" (163).




| | Preamble | Sync | Length | Type | DST Address | SRC Address | Seq Num | Threat Level |
|---|---|---|---|---|---|---|---|---|
| 1 | 0x44 | 0x44 | 84 | 0x0 | 0x30 | 0xa2 | 7397005813 | Threat |
| 2 | 0x14 | 0xcb | 64 | 0x75 | 0xde | 0x46 | 6682479215 | Suspicious |
| 3 | 0x40 | 0x1b | 72 | 0x4c | 0xd8 | 0xf5 | 2024490045 | Normal |
| 4 | 0xbe | 0xa | 47 | 0x48 | 0xb4 | 0x98 | 5325622394 | Threat |
| 5 | 0x1 | 0x1c | 20 | 0xb0 | 0x70 | 0x20 | 2659810389 | Normal |
| 6 | 0x50 | 0x84 | 49 | 0x0 | 0x5e | 0x86 | 4892449975 | Suspicious |
| 7 | 0x98 | 0x5a | 96 | 0x76 | 0x13 | 0x97 | 4114729387 | Threat |
| 8 | 0xac | 0x95 | 23 | 0xd2 | 0x44 | 0x39 | 9385813663 | Suspicious |
| 9 | 0x41 | 0x16 | 85 | 0xb5 | 0x9b | 0x98 | 6084390182 | Normal |

Рисунок 2.13 – Результат аналізу модуля збору трафіку

Результат на рис.2.13 демонструє базову функціональність модуля аналізу, який здатен обробляти дані про мережеві пакети (у цьому випадку - симульовані) та виводити узагальнену інформацію про розподіл рівнів загроз. Користувач може використовувати цю інформацію для отримання загального уявлення про потенційну безпекову ситуацію на основі змодельованих даних. Наступним кроком користувач може натиснути кнопку "Показати графік загроз" для візуального представлення цієї статистики.

Графік на рис.2.14 після натискання на кнопку «Показати графік загроз» наочно відображає статистику загроз, отриману після аналізу симульованих мережевих пакетів. Ми бачимо, що найбільша кількість пакетів була класифікована як підозріла (помаранчевий стовпець є найвищим), трохи менше було нормальних пакетів (зелений стовпець), а кількість загрозливих пакетів (червоний стовпець) є найнижчою серед трьох категорій.

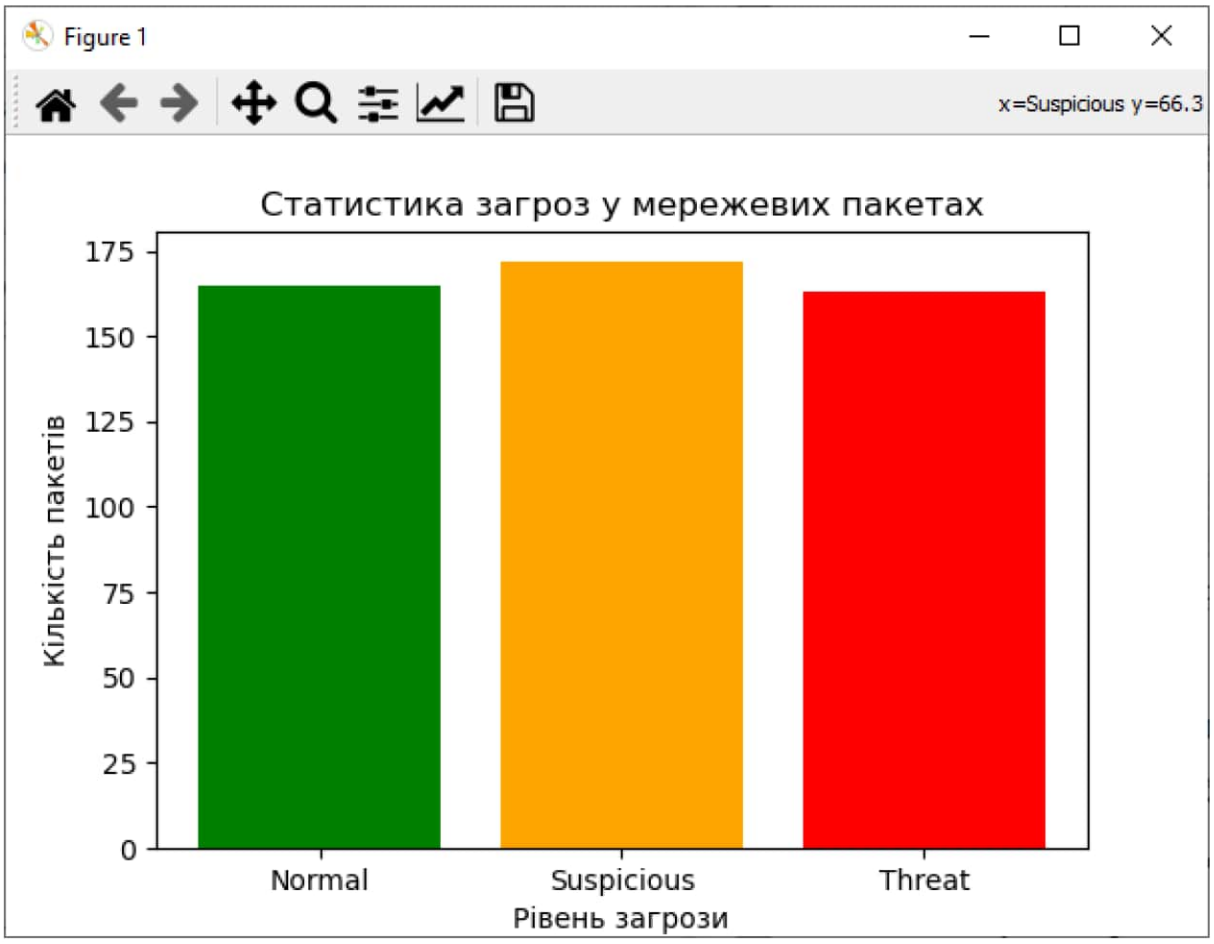



Рисунок 2.14 – Результат статистики виявлених загроз

Користувач натискає кнопку "Візуалізувати трафік". Графік "Трафік у реальному часі" відображає часову динаміку надходження мережевих запитів, класифікованих за трьома основними протоколами: ICMP, UDP та TCP (рис.2.15). Користувач може спостерігати за інтенсивністю трафіку в цілому та за активністю кожного протоколу окремо протягом періоду моніторингу.

В контексті попереднього модуля збору трафіку, який фіксував протоколи пакетів, цей графік є візуальним представленням саме цих даних. Кожна захоплена подія (пакет) відображається на графіку у відповідний момент часу, а її колір вказує на тип протоколу. Аналізуючи графік на рис.2.15, можна виявляти аномальні зміни в інтенсивності трафіку (наприклад, різкі сплески), а також нетипове співвідношення протоколів. Наприклад, під час деяких видів DDoS-атак може спостерігатися значне зростання кількості запитів певного протоколу (наприклад, SYN-флуд - велика кількість TCP-запитів на встановлення з'єднання). У даному ж випадку графік

показує відносно рівномірний та стабільний трафік з представленістю всіх трьох протоколів, що може свідчити про нормальну мережеву активність. Однак, для більш точного висновку потрібен аналіз абсолютної кількості запитів та їхньої поведінки протягом більш тривалого періоду часу.

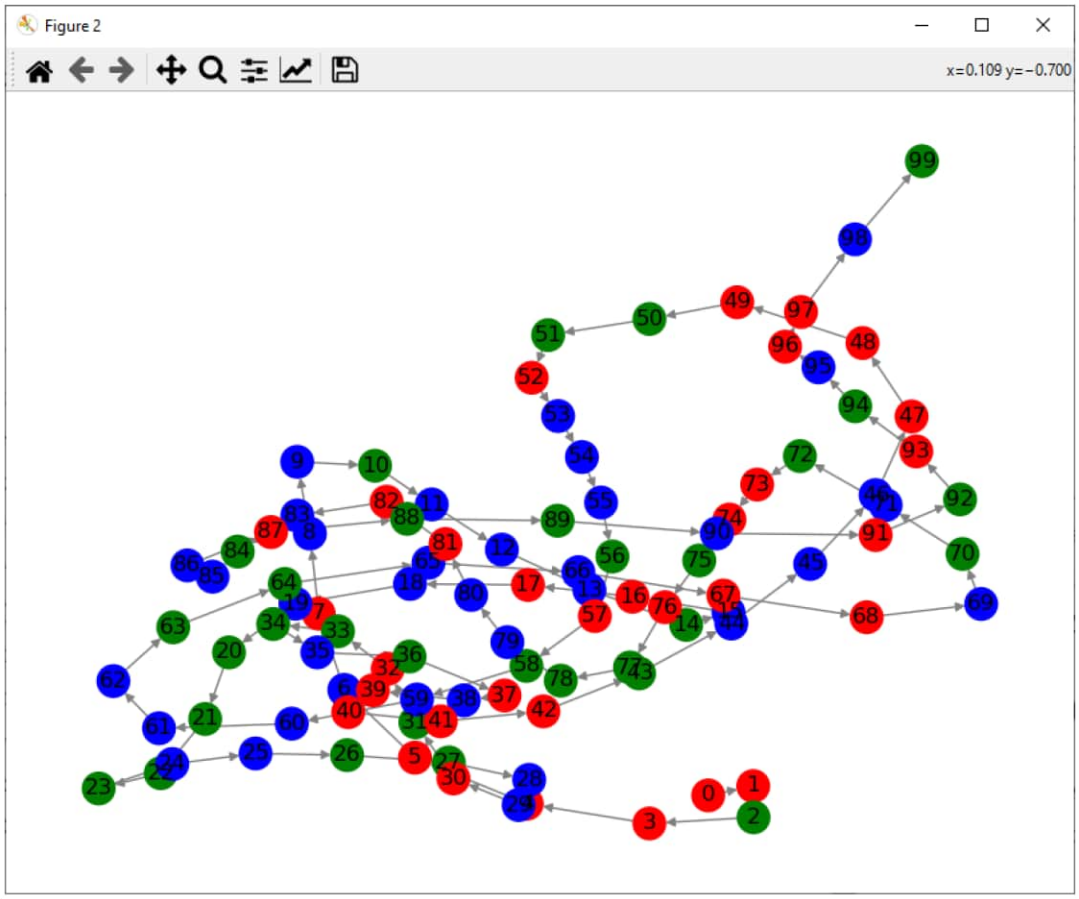



Рисунок 2.15 – Результат "Візуалізувати трафік"

Діаграма варіантів використання для нашого додатку "DDoS Емулятор і Виявлення" являє собою візуальне відображення основних способів взаємодії користувача з системою та ключових функціональних можливостей, які вона надає (рис.2.16). Основним актором у цій діаграмі є "Користувач", який взаємодіє з системою для досягнення різних цілей, пов'язаних з моделюванням та виявленням DDoS-атак.

Система "DDoS Емулятор і Виявлення" обмежена прямокутником, всередині якого розташовані еліпси, що представляють окремі варіанти використання. Кожен варіант використання описує конкретну дію або функціональність, яку користувач може ініціювати. Серед основних варіантів використання, представлених на

діаграмі, є "Запустити атаку", що дозволяє користувачеві імітувати DDoS-атаку з визначеними параметрами для тестування системи виявлення. Варіант використання "Виявити атаку" надає користувачеві можливість запустити процес аналізу зібраного мережевого трафіку з метою виявлення ознак аномальної активності, характерної для DDoS-атак.

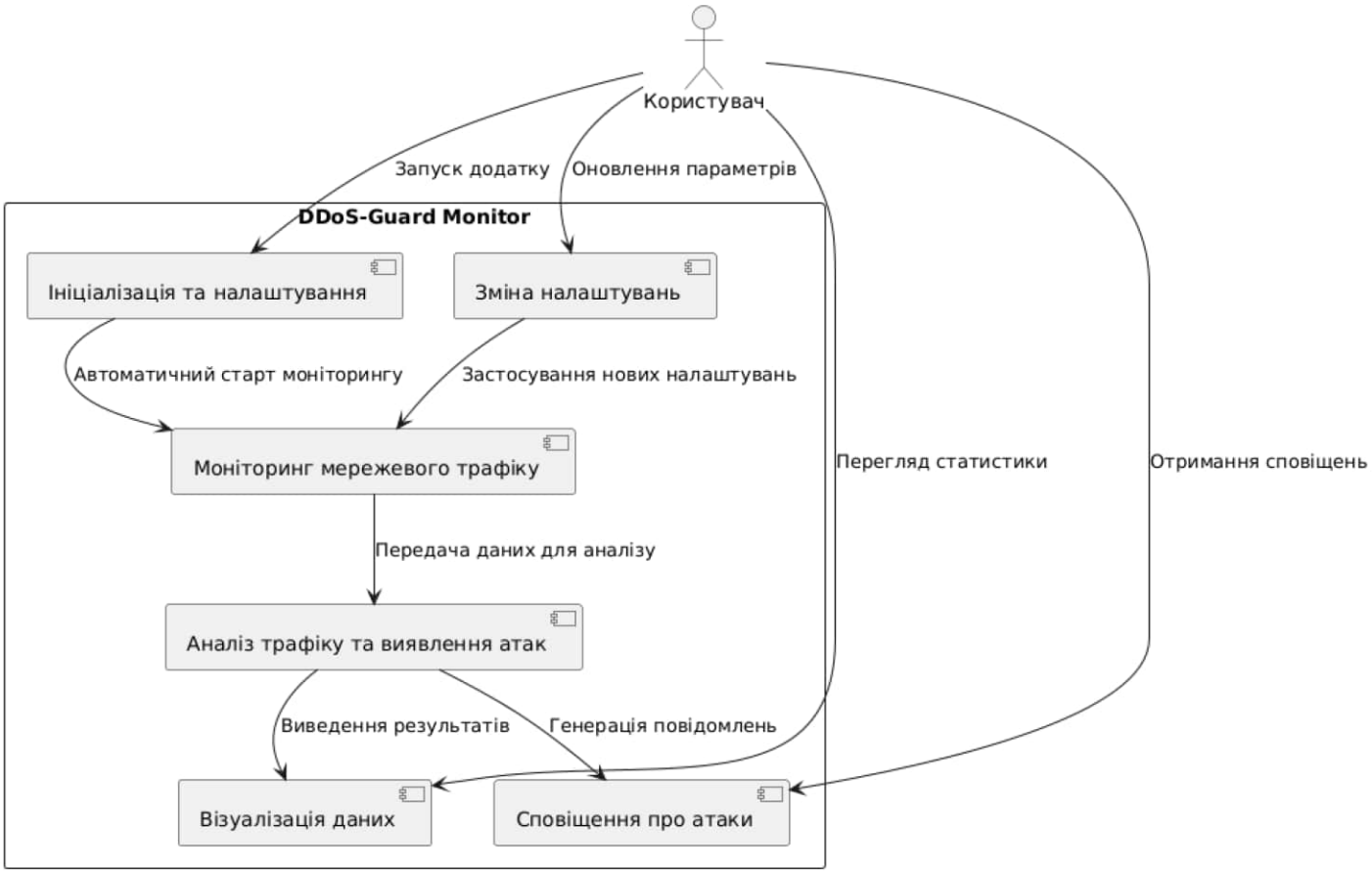



Рисунок 2.16 – Діаграма варіантів використання додатку для моніторингу та виявлення DDoS-атак

Система "DDoS Емулятор і Виявлення" обмежена прямокутником, всередині якого розташовані еліпси, що представляють окремі варіанти використання. Кожен варіант використання описує конкретну дію або функціональність, яку користувач може ініціювати. Серед основних варіантів використання, представлених на діаграмі, є "Запустити атаку", що дозволяє користувачеві імітувати DDoS-атаку з визначеними параметрами для тестування системи виявлення. Варіант використання "Виявити атаку" надає користувачеві можливість запустити процес аналізу

зібраного мережевого трафіку з метою виявлення ознак аномальної активності, характерної для DDoS-атак.

Для спостереження за мережевою активністю в реальному часі передбачено варіант використання "Трафік в реальному часі", який відображає динаміку трафіку з розбивкою за протоколами. Для більш глибокого аналізу зібраних даних користувач може скористатися варіантом використання "Візуалізувати трафік", який надає графічне представлення різних характеристик трафіку. Важливим елементом системи є "Модуль збору трафіку", представлений як варіант використання, що дозволяє користувачеві налаштовувати параметри збору мережевих даних, які є основою для подальшого аналізу та візуалізації. Нарешті, для завершення роботи з додатком передбачено варіант використання "Вийти".

Зв'язки між актором "Користувач" та варіантами використання відображені суцільними лініями зі стрілками, що вказують напрямок ініціації дії. Користувач ініціює запуск атаки, процес виявлення атаки, перегляд трафіку в реальному часі та його візуалізацію, керування параметрами збору трафіку, а також завершення роботи з програмою. Ці зв'язки підкреслюють пряму взаємодію користувача з основними функціональними можливостями системи "DDoS Емулятор і Виявлення".

# ВИСНОВКИ

Результатом проведеної кваліфікаційної роботи стала успішна розробка прикладного програмного забезпечення, призначеного для моніторингу та виявлення DDoS-атак у мережевому трафіку комп’ютерних систем. Для досягнення мети роботи було виконано ряд послідовних завдань, результати яких лягли в основу розробленого прототипу та сформували наступні висновки:

На першому етапі роботи було проведено ґрунтовний аналіз предметної області. Досліджено принципи функціонування DDoS-атак, їхні різноманітні типи (такі як SYN-флуд, UDP-флуд, HTTP-флуд тощо) та характерні особливості. Також було розглянуто сучасні методи виявлення DDoS-атак, включаючи статистичний аналіз аномалій трафіку, сигнатурний аналіз та поведінковий аналіз. Отримані знання стали теоретичною основою для розробки архітектури та вибору методів виявлення в програмному забезпеченні.

На другому етапі було здійснено вивчення існуючих програмних рішень, продуктів та інструментів для моніторингу та виявлення DDoS-атак. Проаналізовано їхні функціональні можливості, переваги та недоліки. Це дозволило визначити актуальні підходи до розробки подібного програмного забезпечення, виявити прогалини в існуючих рішеннях та сформувати вимоги до власного розробленого прототипу.

Третім важливим кроком стала розробка архітектури прикладного програмного забезпечення. Було визначено основні модулі системи, їхні взаємозв'язки та відповідальності. Розроблена архітектура включає модуль збору та аналізу мережевого трафіку, модуль виявлення аномалій на основі обраних методів, а також інтерфейс користувача для відображення даних моніторингу та результатів аналізу. Гнучка та модульна архітектура забезпечує можливість подальшого розширення та вдосконалення системи.

Четвертий етап був присвячений реалізації програмного прототипу. Було розроблено ключові модулі системи, включаючи:

– модуль збору та аналізу мережевого трафіку. Реалізовано функціональність для захоплення мережевих пакетів з обраних інтерфейсів та їхнього базового аналізу, включаючи розбір протоколів та вилучення основних характеристик (IP-адреси, порти, протоколи, розміри пакетів);

– модуль виявлення аномалій. В рамках прототипу було реалізовано базові методи виявлення аномалій, що можуть свідчити про DDoS-атаку. Залежно від обраних методів (наприклад, статистичного аналізу), модуль аналізує зібрані дані на предмет відхилень від нормальних патернів трафіку (наприклад, різке збільшення інтенсивності трафіку з певної IP-адреси або на певний порт);

– інтерфейс користувача. Розроблено простий та інтуїтивно зрозумілий інтерфейс для відображення ключових показників моніторингу (наприклад, графік інтенсивності трафіку в реальному часі, розподіл трафіку за протоколами) та результатів аналізу (виявлені аномалії, потенційні ознаки DDoS-атак).

– На п'ятому етапі було проведено тестування розробленого програмного забезпечення. Тестування здійснювалося на змодельованому мережевому трафіку, що включав як нормальний трафік, так і імітовані DDoS-атаки різних типів. Метою тестування була оцінка ефективності розробленого прототипу у виявленні аномальної активності. Результати тестування показали, що розроблений прототип здатен виявляти деякі базові типи DDoS-атак на основі реалізованих методів аналізу.

У рамках кваліфікаційної роботи було розроблено розподілений програмний додаток, призначений для аналізу та моніторингу ключових мережевих протоколів: ICMP, Traceroute та IPv6 NDP. Метою роботи було створення функціональної системи, яка б дозволяла здійснювати збір даних про стан мережі за допомогою агентів моніторингу, їхню централізовану обробку та візуалізацію для надання користувачеві інформації про працездатність та топологію мережі.

Розроблена архітектура включає в себе розподілених агентів моніторингу, відповідальних за періодичний збір даних за визначеними протоколами, та центральний модуль, який приймає, зберігає та надає ці дані для аналізу та візуалізації. Агенти моніторингу, реалізовані на мові Python, здатні виконувати пінг-

тестування (ICMP), трасування маршруту (Traceroute) та аналіз сусідів IPv6 (NDP). Для забезпечення ефективної передачі даних між агентами та центральним модулем було використано протокол HTTP з форматом даних JSON.

Основний модуль, також розроблений на базі фреймворку Flask на мові Python, забезпечує RESTful API для прийому даних від агентів та надання інформації про зареєстровані агенти та зібрані ними метрики. Для зберігання отриманих даних було спроектовано реляційну базу даних PostgreSQL зі структурованими таблицями для кожного типу моніторингових даних та інформації про агентів. Схема бази даних забезпечує цілісність даних та ефективність виконання запитів.

Для забезпечення базової взаємодії з користувачем було розроблено простий графічний інтерфейс на основі бібліотеки Tkinter. Цей інтерфейс дозволяє ініціювати локальний аналіз PCAP-файлів за протоколами ICMP, Traceroute та IPv6 NDP, надаючи користувачеві можливість переглядати деталі пакетів та основні статистичні дані. Хоча розроблений GUI є базовим, він демонструє потенціал для подальшого розвитку у повноцінний інтерфейс центрального модуля з візуалізацією даних, отриманих від розподілених агентів.

Проведене тестування окремих компонентів розробленої системи підтвердило їхню працездатність. Агенти моніторингу успішно збирають дані за визначеними протоколами, а основний модуль коректно приймає та зберігає цю інформацію. Базовий графічний інтерфейс дозволяє запускати локальний аналіз мережевих даних.

# ПЕРЕЛІК ПОСИЛАНЬ

1. Global DDOS Attack Statistics 2023 [Електронний ресурс] – Режим доступу до ресурсу: https://www.getastra.com/blog/security-audit/ddos-attack-statistics.

2. Закон України «Про загальнообов'язкове державне пенсійне страхування».

3. What is a DDoS (distributed denial of service) attack?. [Електронний ресурс]. – Режим доступу: http:// https://www.ibm.com/topics/ddos.

4. Radware Global Application & Network Security 2016-2017 Report. [Електронний ресурс]. – Режим доступу: https://web.tierpoint.com/radware-ert-2016-2017-report-w.

5. DDoS Attacks. [Електронний ресурс]. – Режим доступу: https://www.imperva.com/learn/ddos/ddos-attacks.

6. Perakovic D. Model for Detection and Classification of DDoS Traffic Based on Artificial Neural Network. / D. Perakovic, M. Perisa, I. Cvitic, S. Husnjak // Telfor Journal, 2017. – Vol. 9, No. 1. – 26–31 pp

7. What Is a Network Packet? [Електронний ресурс] – Режим доступу до ресурсу: https://www.liveaction.com/resources/blog-post/what-is-a-network-packet.

8. Li Y. DDoS attack detection method based on feature extraction of deep belief network. / Y. Li, B. Liu, S. Zhai, M. Chen // IOP Conf. Series: Earth and Environmental Science 252, 2019. – 5 p.

9. Zeinalpour A. Addressing the Effectiveness of DDoS-Attack Detection Methods Based on the Clustering Method Using an Ensemble Method. / A. Zeinalpour, H.A. Ahmed // Electronic, 2022. – No. 11, Article 17. – 16 p.

10. Lopez A. Network Traffic Behavioral Analytics for Detection of DDoS Attacks. / A.D. Lopez, A.P. Mohan, S. Nair // SMU Data Science Review, 2019. – 25 p.

11. Інструмент sklearn.model_selection.GridSearchCV [Електронний ресурс] Режим доступу до ресурсу: https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html.

12. Інструмент cv2 [Електронний ресурс] Режим доступу до ресурсу: https://pypi.org/project/opencv-python/

13. Інструмент sklearn [Електронний ресурс] Режим доступу до ресурсу: https://pypi.org/project/scikit-learn.

14. Інструмент pandas [Електронний ресурс] Режим доступу до ресурсу: https://pypi.org/project/pandas.

# ДОДАТОК А

Текст програми прикладне програмне забезпечення для моніторингу та виявлення DDoS-атак у мережевому трафіку комп’ютерних систем

**Міністерство освіти і науки України**
**НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ**
**“ДНІПРОВСЬКА ПОЛІТЕХНІКА”**

**ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ**

# ПРИКЛАДНЕ ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ДЛЯ МОНІТОРИНГУ ТА ВИЯВЛЕННЯ DDOS-АТАК У МЕРЕЖЕВОМУ ТРАФІКУ КОМП’ЮТЕРНИХ СИСТЕМ

Текст програми

804.02070743.25018-01 12 01

Листів 10

2025

# АНОТАЦІЯ

Дана програма представляє собою програмну систему "DDoS Емулятор", що забезпечує імітацію та виявлення DDoS-атак, а також візуалізацію мережевого трафіку.

Програма призначена для моделювання сценаріїв DDoS-атак та аналізу їх впливу на систему. Вона дозволяє користувачам запускати симуляції атак, виявляти їх на основі заданих параметрів, переглядати трафік у реальному часі та візуалізувати отримані дані, що сприяє розумінню механізмів DDoS-атак та розробці методів протидії. Система підтримує збір та обробку даних трафіку, забезпечуючи повноцінне дослідження.

Програма розроблена як десктопний додаток, що забезпечує її автономну роботу та стабільність. Вона може бути використана для навчальних цілей, досліджень у галузі мережевої безпеки, а також для тестування систем на стійкість до DDoS-атак.

# ЗМІСТ

Main.py

```
import sys
import random
import matplotlib.pyplot as plt
from pyqt5.qtwidgets import qapplication, qmainwindow, qtablewidget, qtablewidgetitem, qvboxlayout, qwidget, \
    qpushbutton, qlineedit, qlabel, qtextedit
from pyqt5.qtgui import qfont


# генеруємо випадкові дані для симуляції мережевих пакетів
def generate_packets(n=500):
    packets = []
    for i in range(n):
        packets.append([
            hex(random.randint(0, 255)),  # preamble
            hex(random.randint(0, 255)),  # sync
            random.randint(10, 100),  # length
            hex(random.randint(0, 255)),  # type
            hex(random.randint(0, 255)),  # dst address
            hex(random.randint(0, 255)),  # src address
            random.randint(1000000000, 9999999999),  # sequence number
            random.choice(["normal", "suspicious", "threat"]),  # threat level
        ])
    return packets


class urhsimulator(qmainwindow):
    def __init__(self):
        super().__init__()
```

```
        self.setwindowtitle("urh simulator")
        self.setgeometry(100, 100, 900, 700)

        self.data = generate_packets()
        self.initui()

    def initui(self):
        layout = qvboxlayout()

        # поле пошуку
        self.search_input = qlineedit(self)
        self.search_input.setplaceholdertext("пошук у пакетах...")
        self.search_input.textchanged.connect(self.search_packets)
        layout.addwidget(self.search_input)

        # таблиця для відображення пакетів
        self.table = qtablewidget()
        self.table.setcolumncount(8)
        self.table.sethorizontalheaderlabels(
            ["preamble", "sync", "length", "type", "dst address", "src address", "seq num", "threat level"])
        layout.addwidget(self.table)

        # віджет для аналізу загроз
        self.threat_analysis = qtextedit(self)
        self.threat_analysis.setreadonly(true)
        self.threat_analysis.setplaceholdertext("натисніть 'запустити аналіз' для перегляду результатів")
        layout.addwidget(self.threat_analysis)
```

```
        # кнопка для запуску аналізу
        self.analyze_button = qpushbutton("запустити аналіз", self)
        self.analyze_button.clicked.connect(self.analyze_threats)
        layout.addwidget(self.analyze_button)

        # кнопка для відображення графіків
        self.graph_button = qpushbutton("показати графік загроз", self)
        self.graph_button.clicked.connect(self.show_graph)
        layout.addwidget(self.graph_button)

        self.load_packets()

        # віджет для компонування
        container = qwidget()
        container.setlayout(layout)
        self.setcentralwidget(container)

    def load_packets(self):
        self.table.setrowcount(len(self.data))
        for row_idx, row_data in enumerate(self.data):
            for col_idx, col_data in enumerate(row_data):
                self.table.setitem(row_idx, col_idx, qtablewidgetitem(str(col_data)))
        self.threat_analysis.clear()  # очищаємо аналіз при запуску програми

    def search_packets(self):
        search_text = self.search_input.text().lower()
        for row_idx in range(self.table.rowcount()):
            row_visible = any(search_text in str(self.table.item(row_idx,
col_idx).text()).lower() for col_idx in
                              range(self.table.columncount()))
```

```
                self.table.setrowhidden(row_idx, not row_visible)

        def analyze_threats(self):
            self.threat_counts = {"normal": 0, "suspicious": 0, "threat": 0}
            for row_idx in range(self.table.rowcount()):
                threat_level = self.table.item(row_idx, 7).text()
                if threat_level in self.threat_counts:
                    self.threat_counts[threat_level] += 1

            analysis_text = f"аналіз загроз:\nнормальні пакети: {self.threat_counts['normal']}\nпідозрілі пакети: {self.threat_counts['suspicious']}\nзагрозливі пакети: {self.threat_counts['threat']}"
            self.threat_analysis.settext(analysis_text)

        def show_graph(self):
            labels = list(self.threat_counts.keys())
            values = list(self.threat_counts.values())
```

Dos.py

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import cv2
import tkinter as tk
from tkinter import messagebox
from sklearn.preprocessing import standardscaler
from sklearn.ensemble import randomforestclassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report


# завантаження даних cic-dos2017
def load_data(filename):
    data = pd.read_csv(filename)
    return data


# попередня обробка даних: вибірки для моделювання
def preprocess_data(data):
    # вибірці даних потрібно очистити від колонок, які не є важливими для класифікації
    features = data.drop(columns=['timestamp', 'label'])  # приклад (вибір без часових міток і міток атаки)
    labels = data['label']  # мітки для класифікації (нормальний трафік або атака)

    # стандартизація даних
    scaler = standardscaler()
    features_scaled = scaler.fit_transform(features)
```

```
    return features_scaled, labels


# навчання моделі для класифікації
def train_model(features, labels):
    # розподіл на навчальну та тестову вибірки
    x_train, x_test, y_train, y_test = train_test_split(features, labels, test_size=0.3, random_state=42)

    # створення моделі randomforestclassifier
    model = randomforestclassifier(n_estimators=100, random_state=42)
    model.fit(x_train, y_train)

    # оцінка моделі
    predictions = model.predict(x_test)
    print(classification_report(y_test, predictions))

    return model


# визуалізація результатів
def visualize_results(data, model):
    # оцінка на всіх даних для виявлення аномалій
    predictions = model.predict(data)

    # створення графіка трафіку
    plt.figure(figsize=(10, 6))
    plt.plot(data, label='трафік')
    plt.title('аналіз мережевого трафіку з використанням cic-dos2017')
```

```
        plt.xlabel('час')
        plt.ylabel('запити')
        plt.legend()

        # збереження графіку
        plt.savefig('traffic_analysis.png')
        plt.close()

        # завантаження графіку через opencv для виявлення аномалій
        img = cv2.imread('traffic_analysis.png')
        cv2.imshow('traffic analysis', img)
        cv2.waitkey(0)
        cv2.destroyallwindows()


    # графічний інтерфейс для емулятора
    class ddosemulatorgui:
        def __init__(self, root):
            self.root = root
            self.root.title("ddos емулятор і виявлення з cic-dos2017")

            self.label = tk.label(root, text="виберіть опцію", font=("helvetica", 14))
            self.label.pack(pady=20)

            self.attack_button = tk.button(root, text="запустити виявлення ddos атаки",
command=self.start_detection)
            self.attack_button.pack(pady=10)

            self.quit_button = tk.button(root, text="вийти", command=root.quit)
            self.quit_button.pack(pady=10)
```

```
            self.model = none

        def start_detection(self):
            # завантаження даних
            filename = "cic-dos2017.csv"  # ваш файл даних
            data = load_data(filename)

            # попередня обробка даних
            features, labels = preprocess_data(data)

            # навчання моделі
            self.model = train_model(features, labels)

            # візуалізація результатів
            visualize_results(features, self.model)


    # головна функція для запуску програми
    def main():
        # створення графічного інтерфейсу
        root = tk.tk()
        gui = ddosemulatorgui(root)
        root.mainloop()


    if __name__ == '__main__':
main()
```