НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ОДЕСЬКА ЮРИДИЧНА АКАДЕМІЯ»

ФАКУЛЬТЕТ КІБЕРБЕЗПЕКИ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Кафедра кібербезпеки

КВАЛІФІКАЦІЙНА РОБОТА

на тему:

«ВИЯВЛЕННЯ ШКІДЛИВОЇ АКТИВНОСТІ MALWARE ЗА ДОПОМОГОЮ АНАЛІЗУ МЕРЕЖЕВОГО ТРАФІКУ»

Виконав: здобувач 4 курсу

Рівень бакалавр

Галузь знань 12 «Інформаційні технології»,

спеціальність 125 «Кібербезпека»

Дрига Артем Вадимович

Керівник: к.т.н., доцент

Бойко Віктор Дмитрович

Рецензент: к.т.н., доцент

Ахмаметьєва Ганна Валеріївна

Одеса – 2024

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ОДЕСЬКА ЮРИДИЧНА АКАДЕМІЯ»
Факультет **кібербезпеки та інформаційних технологій**
Кафедра **кібербезпеки**
Галузь знань: **12 Інформаційні технології**
Спеціальність: **125 Кібербезпека**
Назва освітньої програми: **Кібербезпека**

ЗАТВЕРДЖУЮ
Завідувач кафедри кібербезпеки
професор С.А. Горбаченко
_____ «____ »_____ 20__ року

З А В Д А Н Н Я
на кваліфікаційну (бакалаврську) роботу
здобувачу Дриге Артему Вадимовичу

1. Тема роботи: «Атаки на базові станції мобільного зв'язку: ключові проблеми та шляхи їх вирішення»
Керівник роботи: к.т.н, доцент Віктор Дмитрович Бойко
затверджені наказом НУ ОЮА від «__» _____ 20___ року № _____
2. Строк подання здобувачем роботи «__» _____ 20___ рік
3. Дата видачі завдання«__» _____ 20___ рік

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів бакалаврської роботи	Строк виконання етапів роботи	Примітка

Здобувач _____
(підпис) (прізвище та ініціали)

Керівник роботи _____
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Дрига А. В. Виявлення шкідливої активності malware за допомогою аналізу мережевого трафіку – Кваліфікаційна робота бакалавра за спеціальністю 125 «Кібербезпека», освітньою програмою «Кібербезпека». Кваліфікаційна робота викладена на 32 сторінках основного тексту і 53 сторінках загального тексту, містить 3 розділи, 18 рисунків, 2 додатка та 21 використаних джерел.

Мета дослідження є побудування системи виявлення шкідливої активності в локальних мережах малого та середнього масштабу на основі ідентифікації аномалій у мережевому трафіку.

У кваліфікаційній роботі виконано реалізацію механізмів автоматичного аналізу мережевих пакетів на наявність можливих загроз, інтерфейс для перегляду інцидентів, а також проведено серію симуляцій дій зловмисника з метою протестування створеної системи.

Розроблену систему можна використовувати для невеликих і середніх мереж, наприклад для університетської мережі.

Можливими напрямками подальших досліджень є додавання додаткових можливостей для аналізу мережевого трафіку, створення нових функцій для виявлення інших інцидентів, а також розширення функціональності самої системи.

ІНЦИДЕНТ КІБЕРБЕЗПЕКИ, РЕАГУВАННЯ НА ІНЦИДЕНТИ, PYTHON, SCAPY, DJANGO, МЕРЕЖЕВИЙ ТРАФІК, DOS, ARP-SPOOFING, TCP СКАНУВАННЯ.

ANNOTATION

Dryga A. V. Detection of malicious malware activity using network traffic analysis. – Bachelor's work in Cybersecurity, specialization 125 "Cybersecurity", educational program "Cybersecurity". The thesis consists of 32 pages of main text and 52 pages of total text, comprising 3 chapters, 18 figures, 2 appendices, and references to 21 sources.

The purpose of the study is to build a system for detecting malicious activity in small and medium-scale local networks based on the identification of anomalies in network traffic

In the qualification work, the implementation of mechanisms for automatic analysis of network packets for the presence of possible threats, an interface for viewing incidents, and a series of simulations of the attacker's actions were carried out in order to test the created system.

The developed system can be used for small and medium-sized networks, for example, for a university network.

Possible areas of further research include adding additional capabilities for analyzing network traffic, creating new functions for detecting other incidents, and expanding the functionality of the system itself.

CYBER SECURITY INCIDENT, INCIDENT RESPONSE, PYTHON, SCAPY, DJANGO, NETWORK TRAFFIC, DOS, ARP-SPOOFING, TCP SCANNING.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

DoS – Denial of Service;

DoS - Distributed Denial of Service;

APT - Advanced Persistent Threat;

NIST SP - The National Institute of Standards and Technology Special Publication;

SIEM - Security Information and Event Management;

IDS – Intrusion Detection System;

IDS – Intrusion Prevention System;

PNG - Portable Network Graphics;

PDF - Portable Document Format;

ARP - Address Resolution Protocol;

TCP - Transmission Control Protocol;

IP - Internet Protocol;

HTTP - HyperText Transfer Protocol;

HTML - HyperText Markup Language.

ВСТУП

У наш час, коли цифрові технології проникають в усі сфери життя людини, проблема ефективного забезпечення кібербезпеки стає питанням критичної важливості. Сьогодні вона охоплює не тільки захист комп'ютерних систем, мереж, даних та інформації від несанкціонованого доступу, вторгнень і кібератак, а й захист кінцевих користувачів. У світі, де дедалі вразливішим стає як приватне життя і бізнес-інфраструктура, так і функціонування держави в цілому, необхідність в ефективному захисті даних та інформації стає невід'ємною.

Одним із ключових аспектів кібербезпеки є захист чутливих персональних даних, включно з фінансовою інформацією, медичними записами, а також персональним листуванням. Витік такої інформації може призвести до серйозних наслідків, таких як фінансові втрати, загрози репутації та потенційні порушення особистого життя.

Одним із головних векторів атак на інформаційні системи є комп'ютерні мережі, які з'єднують окремі пристрої, компанії та навіть країни, даючи змогу здійснювати комунікування та іншим чином взаємодіяти людям у різних частинах світу. Тому вони залишаються головним каналом передачі інформації, зокрема й приватної. Це робить комп'ютерні мережі важливою ціллю для атак зловмисників, а тому їхній захист - одне з основних завдань спеціалістів з кібербезпеки.

Особливо важливим питання кібербезпеки стало для нашої країни, яка щодня зазнає атак агресора не тільки у фізичному просторі, а й у віртуальному. Лише торік Україна зіткнулася з 2543 кіберінцидентами, що на 16% більше, ніж у 2022 році [1]. Таким чином, оперативне виявлення таких інцидентів є запорукою нормального функціонування не лише бізнесу, а й держави загалом.

Критично вразливими залишаються мережі маленького і середнього розмірів, оскільки їх, найчастіше, адмініструє мала кількість адміністраторів. Невеликі розміри робить розгортання ресурсовитратних інструментів для

виявлення інцидентів неефективним, водночас невеликий контингент адміністраторів унеможливлює ручне виявлення зловмисної діяльності.

Мета дослідження: побудувати систему виявлення шкідливої активності в локальних мережах малого та середнього масштабу на основі ідентифікації аномалій у мережевому трафіку.

Для досягнення мети було сформульовано такі завдання:

1. Огляд прикладів найбільш поширеної шкідливої активності;
2. Визначення характерних ознак шкідливої мережевої активності;
3. Планування системи аналізу мережевого трафіку для виявлення шкідливої активності;
4. Планування веб-інтерфейсу для системи аналізу мережевого трафіку;
5. Побудова системи аналізу мережевого трафіку;
6. Тестування системи аналізу мережевого трафіку на прикладах поширених атак.

Об'єкт дослідження: мережевий трафік локальних мереж малого та середнього масштабу.

Предмет дослідження: виявлення ознак компрометації в мережевому трафіку.

Реалізація даної системи дасть змогу отримати зручний інструмент виявлення аномалій і запобігання зловмисній активності в локальних мережах малого та середнього розміру, що істотно підвищить їхню захищеність.

Основні висновки роботи були опубліковані та апробовані на наукових конференціях «Кіберпростір в умовах війни та глобальних викликів XXI століття: теорія та практика» та «Інформаційне суспільство: проблеми та перспективи» [2] [3].

1 АНАЛІЗ МЕРЕЖЕВОГО ТРАФІКУ ЯК СКЛАДОВА ДІЯЛЬНОСТІ З РЕАГУВАННЯ НА ІНЦИДЕНТИ

1.1 Процес реагування на інциденти

У 2017 року в Україні було зафіксовано одну з наймасштабніших кібератак за весь час - вірус Petya заразив тисячі комп'ютерів по всій країні, зашифрувавши всі дані на них і вимагаючи за розблокування викуп. Хоча географія ураження Petya не була обмежена тільки Україною, так як протягом доби після розповсюдження воно поширилося в усьому світі, але кількість інфікованих комп’ютерів лише в Україні склала не менше 12500, а постраждали від атаки державні та комунальні установи, об’єкти критичної інфраструктури, мобільні оператори та банки [4]. Це показує, що жодна компанія не застрахована від можливих кібератак, а наслідки від їх реалізації можуть бути катастрофічними. Навіть найкращу систему захисту можна обійти, а тому фахівцям з кібербезпеки завжди потрібно бути готовим до будь-яких інцидентів.

Інцидент - подія, яка фактично або потенційно загрожує конфіденційності, цілісності або доступності інформаційної системи або інформації, яку система обробляє, зберігає або передає, або яка є порушенням чи неминучою загрозою порушення політики безпеки, процедур безпеки або політики прийнятного використання [5].

Приклади можливих інцидентів:

- Використання шкідливого програмного забезпечення для атаки на систему;
- Несанкціоновані спроби доступу до систем або даних;
- DoS та DDoS атаки (наприклад, SYN-flood атака);
- Розвідка мережі (наприклад, сканування портів);
- Атака типу Man-In-The-Middle (наприклад, ARP Spoofing);
- APT-атаки;
- Фішингові атаки.

Компанії щодня зустрічаються з десятками, якщо не сотнями спроб зловмисного проникнення в систему, тому важливо з самого спочатку правильно налаштувати процес реагування на інциденти в організації. Це допоможе максимально швидко та ефективно протистояти можливим порушенням, не дасть порушникам закріпитися в системі, а також зменшить можливі ризики для компанії.

Реагування на інциденти - це стратегічний процес, який компанії, зокрема IT-відділи та команди розробників, здійснюють для швидкого реагування на незаплановані події або перебої в обслуговуванні. Його мета - відновити операційну функціональність і зменшити потенційні збитки, спричинені кіберзагрозами або порушеннями [6].

Згідно зі спеціальною публікацією NIST SP 800-61 "Computer Security Incident Handling Guide" [7], процес реагування на інциденти поділяється 4 етапи (рисунок 1.1):

1. Підготовка (Preparation): процес планування. Під час цього етапу вживаються заходи для забезпечення наявності необхідних інструментів і ресурсів;
2. Виявлення та аналіз (Detection and analysis): процес виявлення та оцінки. Під час цього етапу створюються процеси для виявлення та оцінки інцидентів;
3. Локалізація, ліквідація та відновлення (Containment, eradication, and recovery): процес мінімізації та пом'якшення наслідків. Під час цього етапу спеціалісти з кібербезпеки та інші співробітники співпрацюють, щоб мінімізувати вплив інциденту та пом'якшити будь-які операційні перебої;
4. Діяльність після інциденту (Post-incident activity): процес навчання. Під час цього етапу впроваджуються нові протоколи та процедури з метою допомогти зменшити кількість подібних інцидентів у майбутньому.

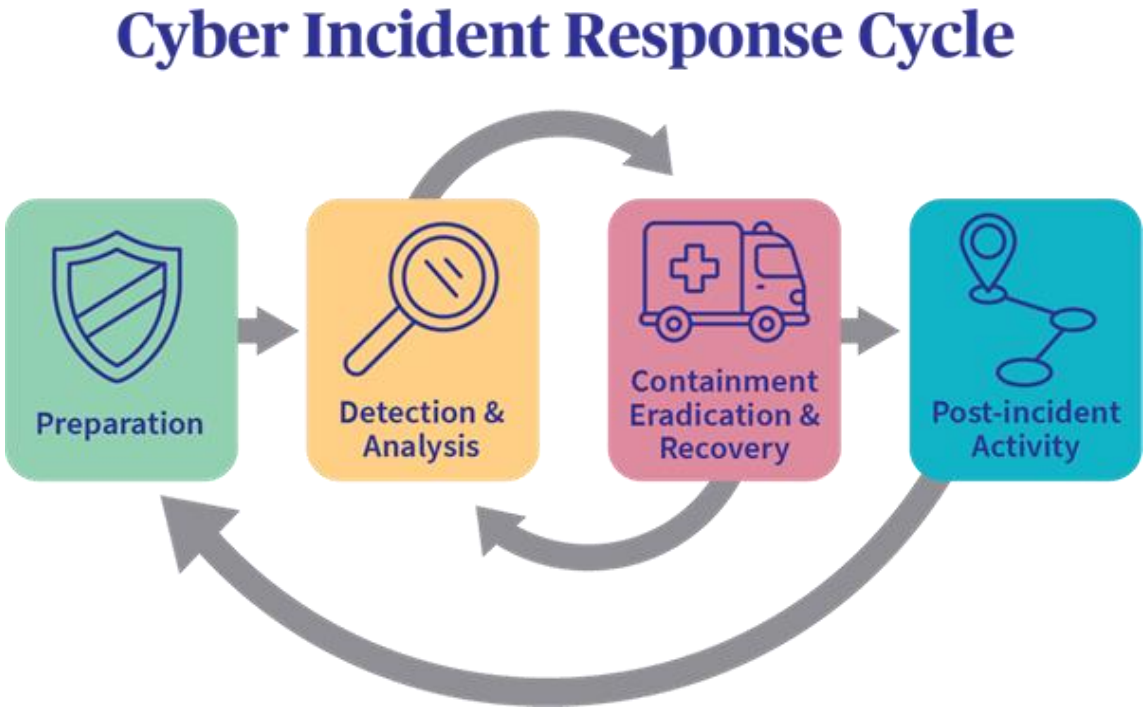

Рисунок 1.1 – Процес реагування на інциденти

1.2 Інструменти реагування на інциденти

Існує велика кількість різних інструментів для реагування на інциденти, які різняться сферою застосування. Наприклад, для запобігання факту виникнення інциденту внаслідок експлуатації вразливостей системи використовуються сканери вразливостей.

Найпопулярнішими та найефективнішими залишаються ті інструменти, які виявляють інцидент під час того як він відбувається. Для того, щоб організації могли захистити свої системи від атак, вони повинні знати про будь-які ознаки вторгнення. Інструменти виявлення інформують фахівців з безпеки про активність, що відбувається в мережі або системі. Вони роблять це шляхом безперервного моніторингу мереж і систем на предмет будь-якої підозрілої активності. Як тільки виявляється щось незвичне або підозріле, інструмент активує сповіщення або вживає дії для усунення загрози. Прикладами таких інструментів являються EDR, XDR, MDR, антивіруси тощо. З усього списку інструментів виявлення можливих інцидентів особливо затребуваними залишаються інструменти виявлення аномалій у мережевому трафіку.

Саме мережевий трафік є первинним джерелом для виявлення потенційних атак. Його аналіз дає змогу швидко реагувати на загрози та вживати заходів щодо їх запобігання, мінімізуючи потенційні поширення шкідливого програмного забезпечення та переривання роботи мережі.

При цьому, незалежно від розміру мережі, в будь-який момент часу в ній може бути величезний обсяг мережевого трафіку, в якому потрібно знайти ознаки загроз. Ручне виявлення індикаторів порушень у навіть маленькій мережі займає велику кількість часу. З цієї причини постійно використовуються такі інструменти, як SIEM, IDS, IPS і аналізатори мережевих пакетів.

IDS (Intrusion Detection System) – інструмент, який спостерігає за мережевим трафіком на предмет зловмисних транзакцій і негайно надсилає сповіщення, коли її виявлено, таким перевіряючи мережу або систему на наявність зловмисних дій або порушень політики. Існує декілька підвидів IDS. Для аналізу мережі використовують NIDS (Network-Based Intrusion Detection System), встановлюються в запланованій точці мережі для перевірки трафіку з усіх пристроїв в мережі (рисунок 1.2). Вона виконує спостереження за трафіком, що проходить по всій підмережі, і порівнює трафік, який передається по підмережах, з колекцією відомих атак [8].

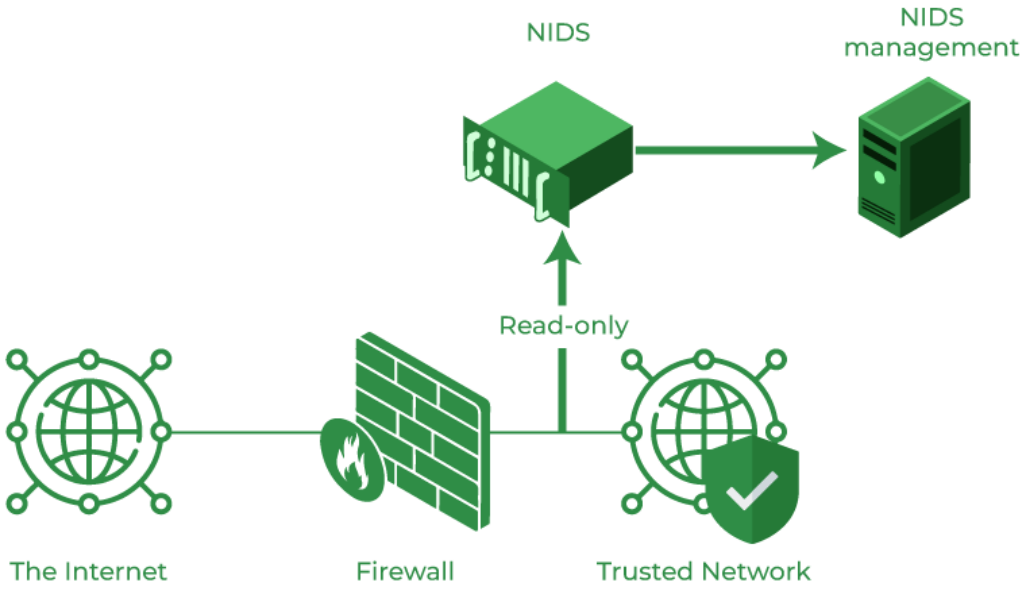

Рисунок 1.2 – Архітектура мережі разом з NIDS

Основним недоліком IDS та NIDS є часті хибнопозитивні спрацьовування, тобто інтерпретації легітимного трафіку як шкідливого. Крім цього, впровадження NIDS може призвести до проблем з продуктивністю, оскільки він повинен аналізувати великий обсяг даних у режимі реального часу.

IPS (Intrusion Prevention System) - інструмент, який відстежує мережеву або системну активність на предмет зловмисної діяльності. Основними функціями IPS є виявлення шкідливої активності, збір інформації про неї, повідомлення про неї та спроба заблокувати або зупинити її. І IPS, і IDS відстежують мережевий трафік і системну активність на предмет зловмисної діяльності, але IPS при цьому намагається заблокувати можливу загрозу, на відміну від IDS. Для IPS також існують різні категорії, і для аналізу мережевого трафіку вирізняють NIPS (Network-Based Intrusion Prevention System), який встановлюється на периметрі мережі і контролює весь трафік, який входить і виходить з мережі (рисунок 1.3) [9].

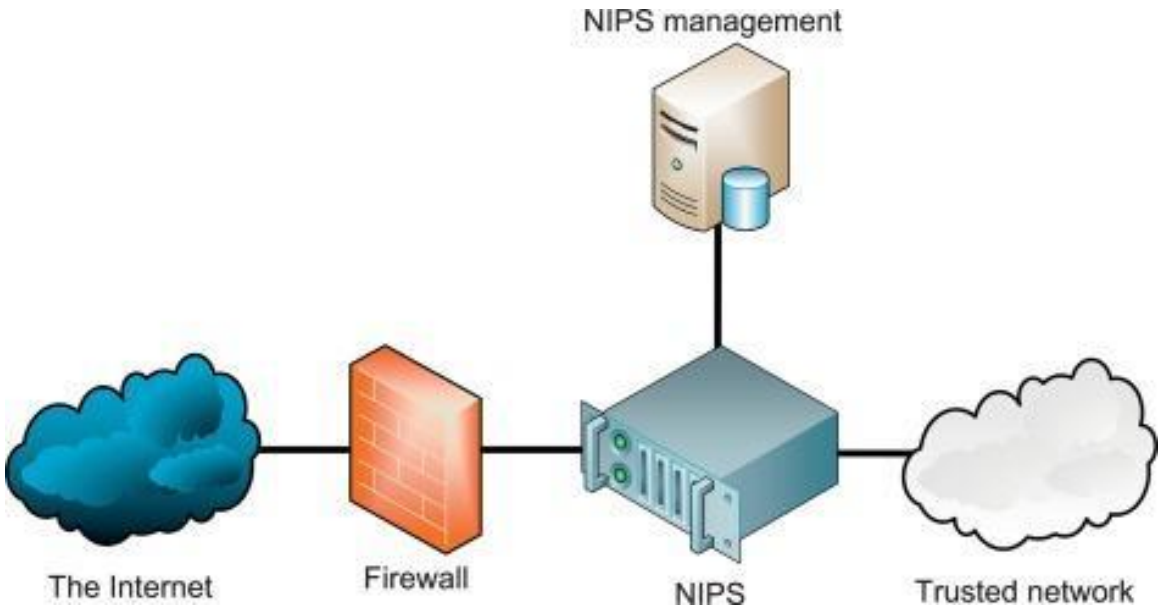

Рисунок 1.3 – Архітектура мережі разом з NIPS

IPS і NIPS властиві такі самі недоліки, як і для IDS і NIDS, водночас додаються нові. З огляду на те, що IPS не тільки повідомляє про можливий інцидент, а й блокує його, він може порушити роботу мережі, якщо блокуватиме трафік у разі хибнопозитивних спрацьовувань. Також у разі порушення роботи або

відключення IPS також перестане працювати і мережа. На рисунку 1.4 представлено різницю в проєктуванні мережі під час використання IDS і IPS.

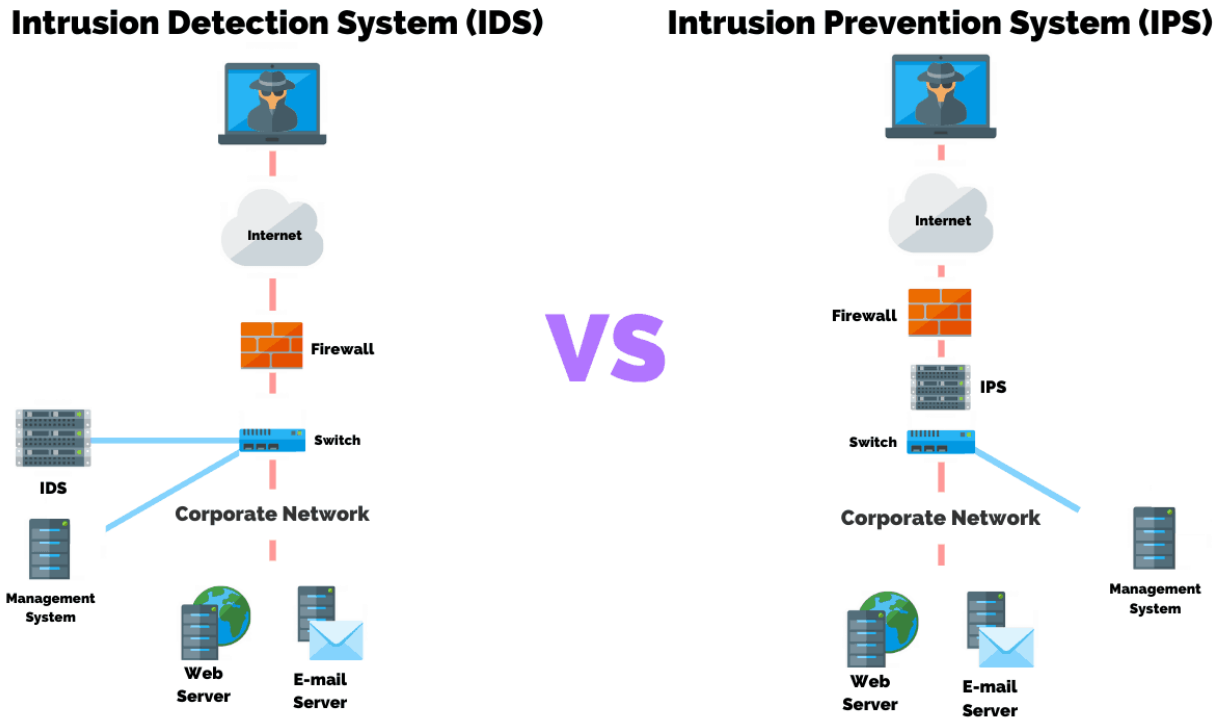

Рисунок 1.4 – Різниця між IDS та IPS

SIEM (Security Information and Event Management) – інструмент, який допомагає керувати інцидентами безпеки, збираючи та аналізуючи дані журналів, події безпеки та джерела даних. Спеціалісти з кібербезпеки використовують SIEM для управління інцидентами безпеки, а також для швидкого виявлення та реагування на потенційні загрози.

SIEM працює шляхом збору даних про безпеку з різних джерел у мережі організації, таких як журнали подій мережевих пристроїв, IDS та IPS, серверів і додатків. Ці дані збираються в центральному сховищі, де вони проходять кореляційний аналіз для виявлення закономірностей, аномалій і потенційних інцидентів безпеки [10].

SIEM виконує наступні операції:

1. Збирає дані з різних джерел;
2. Нормалізує і проводить агрегацію зібраних даних;
3. Нормалізовані дані аналізуються і корелюються за допомогою заздалегідь визначених правил, щоб виявити взаємозв'язок між ними.

4. Повідомляє про виявлені аномалії.

На рисунку 1.5 зображено приклад SIEM системи.

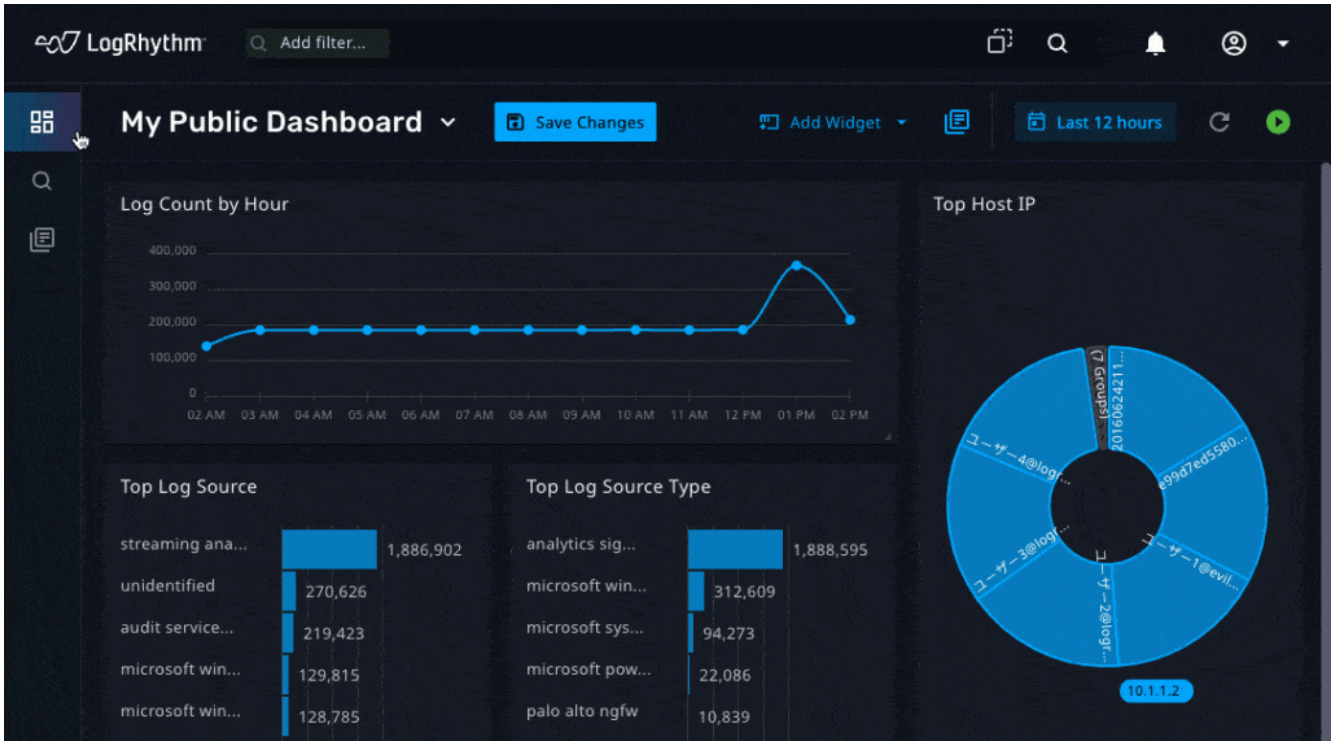

Рисунок 1.5 – Приклад SIEM системи

Попри всі можливості SIEM систем, вони досить вимогливі до ресурсів, оскільки зберігають величезну кількість журналів подій, а також вимагають спеціалізованих знань для налаштування кореляційних правил.

Особливо ефективним поєднанням є комбінація IDS і SIEM інструментів - поки IDS виявляє можливі інциденти і відправляє їх SIEM системі, остання аналізує отримані дані та візуалізує їх. Однак через складність налаштування та підвищену вимогливість до ресурсів, така комбінація не підходить для невеликих мереж.

Аналізатор мережевих пакетів (або сніфер) - інструмент, що призначений для перехоплення та аналізу трафіку даних у мережі. Окрім використання у сфері кібербезпеки як інструменту для моніторингу мереж та виявлення підозрілої активності, аналізатори мережевих пакетів можна використовувати для збору мережевої статистики, наприклад, пропускної здатності або швидкості, а також для усунення проблем з продуктивністю мережі, наприклад, уповільнення роботи.

Прикладами аналізаторів мережевих пакетів є tcpdump, який працює в консольному режимі, та Wireshark, який має зручний графічний інтерфейс (рисунок 1.6).

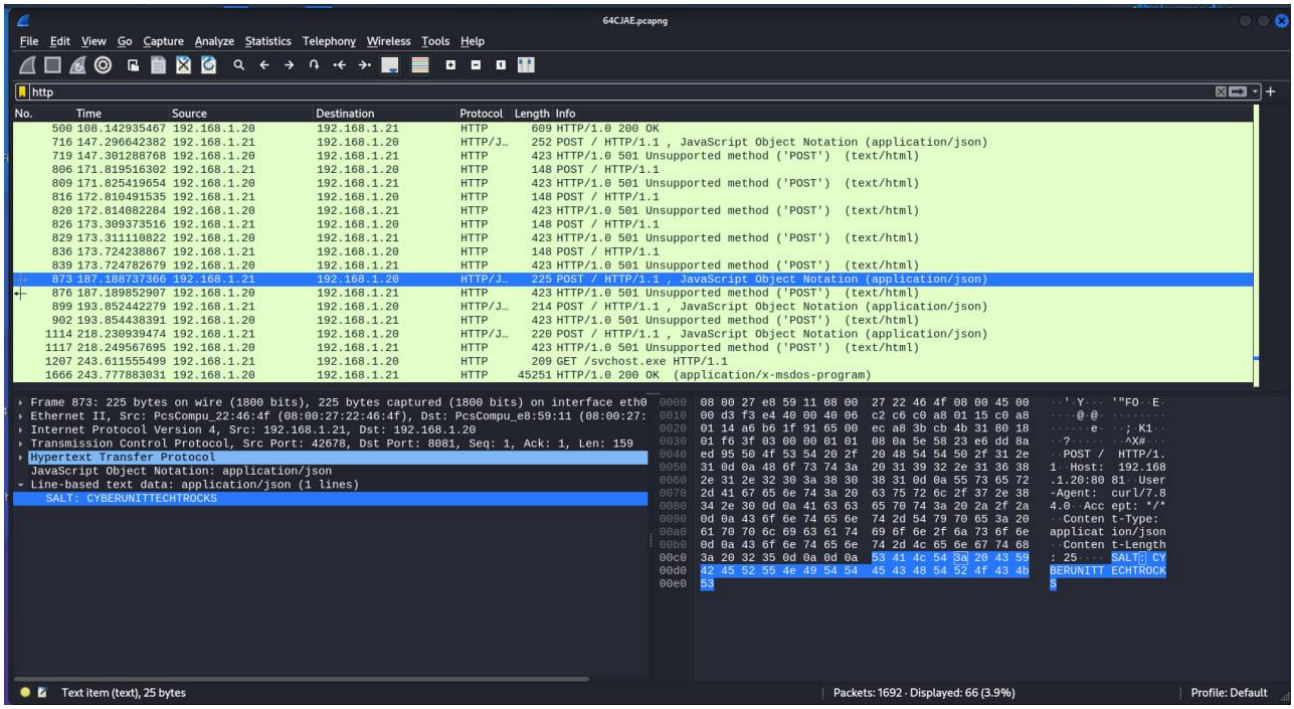

Рисунок 1.6 – Графічний інтерфейс Wireshark

Wireshark і tcpdump є стандартними інструментами під час реагування на інциденти. Однак, вони підходять не для всіх випадків [8]. Наприклад, неефективно використовувати Wireshark з великим об'ємом мережевого трафіку через його особливість - для роботи з файлом йому потрібно виділити в пам'яті приблизно в десять разів більше місця, ніж фактичний його розмір.

Також ці інструменти хоча й зручно показують інформацію про мережеві пакети, але потребують постійного керування адміністратором мережі та не працюють в автоматичному режимі.

Проаналізувавши всі інструменти для виявлення інцидентів у мережевому трафіку, було прийнято рішення розробити власну платформу, головними особливостями якої були б:

- Автоматичний аналіз мережевих пакетів на наявність можливих загроз;
- Можливість додавання нових правил виявлення загроз;

- Інтерфейс для перегляду інцидентів;
- Можливість роботи з файлами великих розмірів;
- Споживання незначних ресурсів системи;
- Невтручання у звичайну роботу мережі.

Для початку розробки платформи потрібно визначитися з технологіями, які будуть задіяні, а також проаналізувати варіанти шкідливої діяльності.

2 ПІДГОТОВКА ДО РОЗРОБКИ СИСТЕМИ ДЛЯ АНАЛІЗУ МЕРЕЖЕВОГО ТРАФІКУ

2.1 Огляд використовуваних технологій та інструментів

Для створення платформи для аналізу мережевого трафіку потрібно насамперед вибрати мову програмування. Була обрана інтерпретована, об'єктно-орієнтована мова програмування високого рівня Python, яка:

- Ефективно обробляє файли великих об'ємів;
- Має зручний інструментарій для роботи з мережевим трафіком у вигляді бібліотек;
- Дозволяє відносно легко і швидко додавати нові правила для виявлення аномалій у мережевому трафіку.

Як вже було зазначено, Python має велику кількість бібліотек, які дають змогу розбирати мережевий трафік і працювати безпосередньо з мережевими пакетами. Найпопулярнішими такого типу бібліотеками є Scapy і PyShark. З метою обрати відповідний інструмент для аналізу мережевого трафіку, було проведено кілька тестів, завданням яких було порівняння швидкості Scapy і Pyshark на прикладі відкриття файлів різних розмірів. Для візуалізації отриманих результатів тестування на продуктивність було використано бібліотеку matplotlib [11].

Бібліотека Scapy має у своєму розпорядженні кілька функцій, які дають змогу зчитувати дані з файлів із мережевим трафіком:

1. rdpcap();
2. PcapReader();
3. sniff(offline="шлях до файлу").

Функції rdpcap() та sniff(offline="шлях до файлу") зчитують файли ціликом, в той PcapReader() зчитує пакети по одному.

Pyshark містить тільки одну функцію для зчитування пакетів з файлу – FileCapture().

Для проведення тестування для кожної функції було поставлено завдання відкрити по десять разів два файли, отримані результати були записані у файл

(рисунок 2.1), після чого за їх допомогою було побудовано графіки (рисунок 2.2). Весь код для порівняння двох бібліотек наведено в додатку А.

```
Час відкриття ARP файлу за допомогою rdpcap 2.9349801199999774
Час відкриття ARP файлу за допомогою rdpcap 2.918763772000034
Час відкриття ARP файлу за допомогою rdpcap 2.9368517620000603
Час відкриття ARP файлу за допомогою rdpcap 3.0045653919999904
Час відкриття ARP файлу за допомогою rdpcap 2.9039677320000069
Час відкриття ARP файлу за допомогою rdpcap 3.09767298700001
Час відкриття ARP файлу за допомогою rdpcap 2.9730261930000097
Час відкриття ARP файлу за допомогою rdpcap 3.011191957000051
Час відкриття ARP файлу за допомогою rdpcap 3.152111665999996
Час відкриття ARP файлу за допомогою rdpcap 3.0320508700000346
Час відкриття DoS файлу за допомогою rdpcap 2.9417928589999747
Час відкриття DoS файлу за допомогою rdpcap 2.9348551140000154
Час відкриття DoS файлу за допомогою rdpcap 2.9674909120000166
Час відкриття DoS файлу за допомогою rdpcap 3.0292930360000128
Час відкриття DoS файлу за допомогою rdpcap 3.018072529000051
Час відкриття DoS файлу за допомогою rdpcap 3.6102605519999997
Час відкриття DoS файлу за допомогою rdpcap 3.359470965000014
Час відкриття DoS файлу за допомогою rdpcap 3.2173626659999854
Час відкриття DoS файлу за допомогою rdpcap 3.037454503999925
Час відкриття DoS файлу за допомогою rdpcap 3.0022116830000414
Час відкриття ARP файлу за допомогою Pyshark 0.0004898999999975482
Час відкриття ARP файлу за допомогою Pyshark 0.0004916840000532829
Час відкриття ARP файлу за допомогою Pyshark 0.0004522970000 3801554
Час відкриття ARP файлу за допомогою Pyshark 0.0004508460000351988
Час відкриття ARP файлу за допомогою Pyshark 0.0004237880000 3461947
Час відкриття ARP файлу за допомогою Pyshark 0.0004475590000836846
Час відкриття ARP файлу за допомогою Pyshark 0.0004802060000201891
Час відкриття ARP файлу за допомогою Pyshark 0.0004358649999858244
Час відкриття ARP файлу за допомогою Pyshark 0.0004335630000141346
Час відкриття ARP файлу за допомогою Pyshark 0.0004334089999247226
Час відкриття DoS файлу за допомогою Pyshark 0.0004153029999542923
Час відкриття DoS файлу за допомогою Pyshark 0.0003952200000867378 5
Час відкриття DoS файлу за допомогою Pyshark 0.0004145919999700709
Час відкриття DoS файлу за допомогою Pyshark 0.0004252569999607658
Час відкриття DoS файлу за допомогою Pyshark 0.0004088069999858 4506
Час відкриття DoS файлу за допомогою Pyshark 0.0004328500000 3834284
Час відкриття DoS файлу за допомогою Pyshark 0.0004456550002136105
Час відкриття DoS файлу за допомогою Pyshark 0.0004384679999702712
Час відкриття DoS файлу за допомогою Pyshark 0.0004658240000026126
Час відкриття DoS файлу за допомогою Pyshark 0.0004052329999240101
Час відкриття ARP файлу за допомогою PcapReader 0.0001178100000061022
Час відкриття ARP файлу за допомогою PcapReader 9.098700002141413e-05
Час відкриття ARP файлу за допомогою PcapReader 8.818700007395819e-05
```

Рисунок 2.1 – Приклад записаних даних у файл

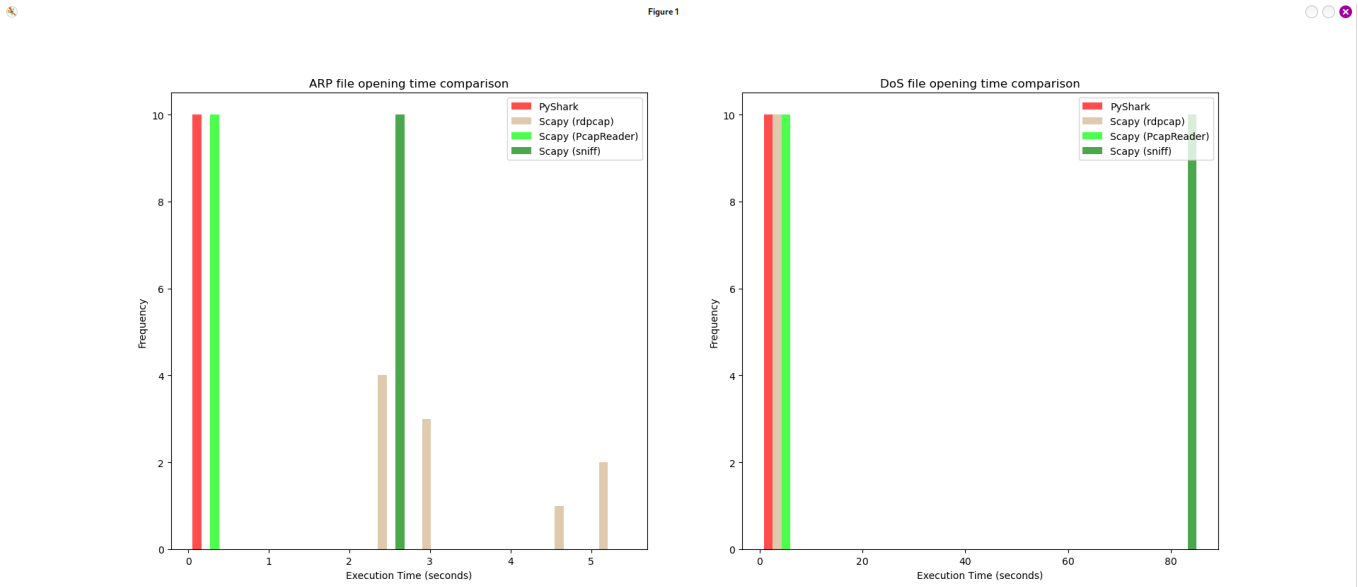

Рисунок 2.2 – Графічний інтерфейс Wireshark

Тести показали, що функція PcapReader() бібліотеки Scapy і функція FileCapture() бібліотеки PyShark працюють приблизно однаково. Крім цих функцій, Scapy також має у своєму розпорядженні безліч корисних інструментів, які можуть спростити життя системним адміністраторам, мережевим інженерам і фахівцям з кібербезпеки, тому було обрано цю бібілотеку [2]. Наприклад, Scapy може зібрати всю інформацію про шлях TCP пакета і на основі цього створити карту його маршруту.

Ще одним прикладом вбудованого функціоналу Scapy є те, що він дає змогу за допомогою кількох рядків коду малювати діаграми, що візуалізують структуру мережевих пакетів, а також зберігати їх у різних форматах, таких як PNG, PostScript і PDF. Нижче наведено приклад програми для такої візуалізації, а на рисунку 2.3 наведено саму діаграму.

```
a=rdpcap("/spare/captures/isakmp.cap")
a[423].pdfdump(layer_shift=1)
a[423].psdump("/tmp/isakmp_pkt.eps",layer_shift=1)
```

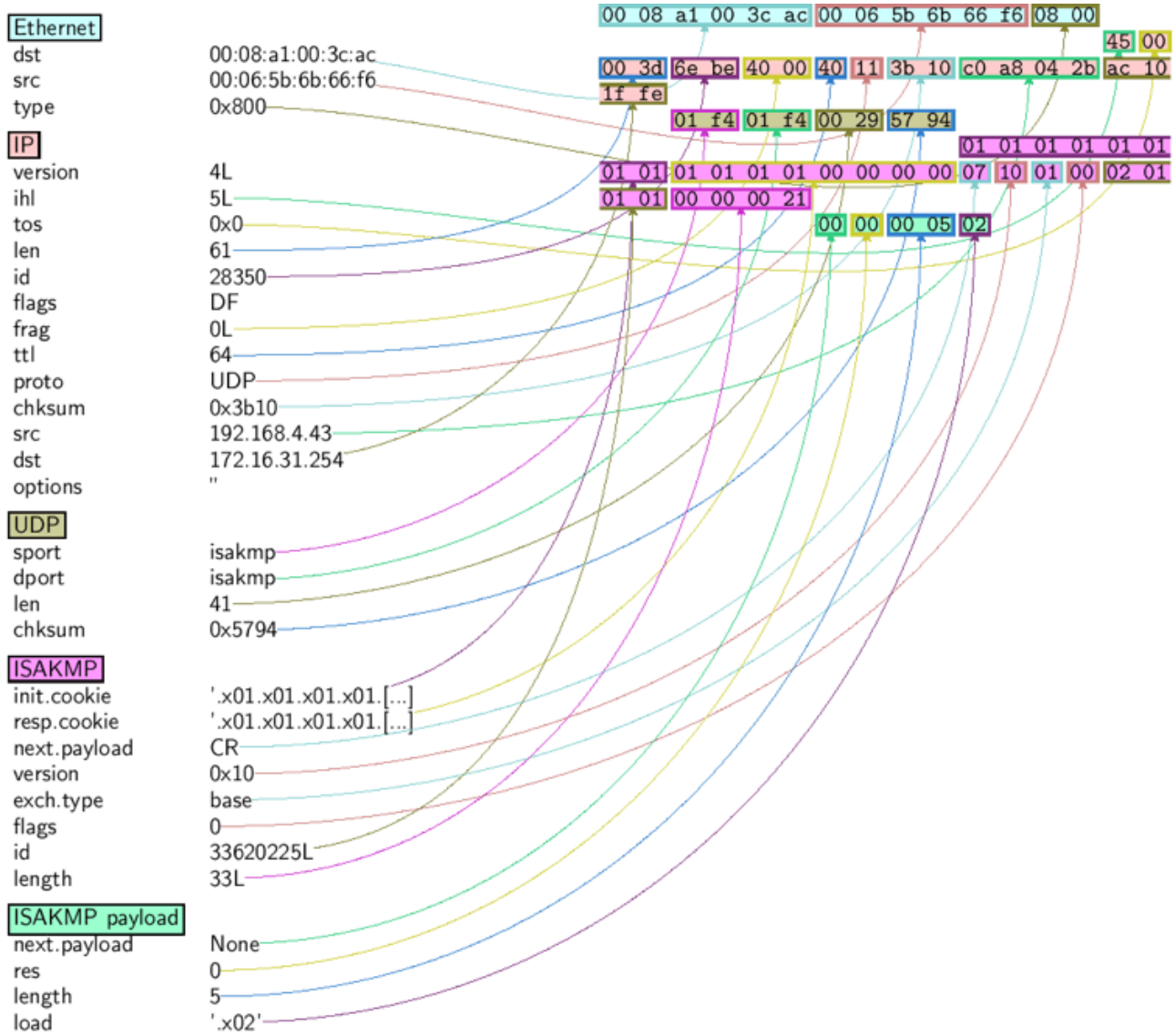

Рисунок 2.3 – Приклад діаграми

Після вибору бібліотеки для роботи з мережевим трафіком, потрібно вибрати інструмент, який дасть змогу відображати результати аналізу. Існують різні варіанти інтерфейсу, наприклад:

1. Графічний інтерфейс користувача;
2. Текстовий інтерфейс користувача;
3. Інтерфейс командного рядка;
4. Веб-інтерфейс.

Було обрано веб-інтерфейс для вирішення завдання з відображенням даних, оскільки такий тип інтерфейсу дасть змогу отримати доступ до системи звідки завгодно. Для мови програмування Python існує кілька веб-фреймворків, які дадуть змогу написати веб-інтерфейс. Серед найпопулярніших це Flask, Django і Pyramid. Django має вже готову панель адміністратора, яка дасть змогу на даному етапі переглядати результати аналізу мережевого трафіку і не витрачати час на створення зовнішнього вигляду інтерфейсу, а зосередитися на написанні логіки роботи.

Також Django має в собі безліч готових додаткових модулів, які в майбутньому можуть стати в пригоді під час розширення системи. Такі модулі постійно додаються та оновлюються.

2.2 Ознаки шкідливої діяльності в мережевому трафіку

Як уже було зазначено в першому розділі, серед найпоширеніших видів мережевих атак є DoS та DDoS атаки та атаки типу Man-In-The-Middle.

Так, кількість DoS та DDoS атак в 2023 році на фінансовий сектор зросла на 78% у порівнянні з 2022 роком, а на урядові служби – на 108% [12]. У той же час, в першому кварталі 2023 року кількість атак типу Man-In-The-Middle на поштові скриньки зросла на 35% порівняно з аналогічним періодом 2022 року [13]. Зростання популярності робить важливим завданням визначення такого роду атак.

Атака типу Man-In-The-Middle (MITM) - це загальний термін для позначення ситуації, коли зловмисник позиціонує себе в розмові між користувачем і додатком - або для підслуховування, або для того, щоб видати себе за одну зі сторін, створюючи враження, що відбувається звичайний обмін інформацією. Метою атаки є викрадення особистої інформації, такої як облікові дані для входу в систему, реквізити рахунків і номери кредитних карток. Цілями зазвичай стають

користувачі фінансових додатків, сайтів електронної комерції та інших веб-сайтів, де потрібен вхід в систему. Інформація, отримана під час атаки, може бути використана для багатьох цілей, включаючи крадіжку особистих даних, несанкціоновані перекази коштів або незаконну зміну пароля [14].

Приклад такого роду атаки - ARP-spoofing, яка ґрунтується на недоліках протоколу ARP.

ARP-spoofing проводиться в кілька етапів (рисунок 2.4):

1. Спочатку зловмисник надсилає ARP-відповідь комп'ютеру жертви, де вказує, що його MAC-адреса відповідає IP-адресі маршрутизатора;
2. Комп'ютер жертви оновлює свою ARP-таблицю згідно з отриманою відповіддю;
3. Тепер весь мережевий трафік жертви перед тим, як відправитися на маршрутизатор, потрапляє до рук зловмисника, який може його прочитувати і модифікувати;
4. Ті ж самі кроки зловмисник може виконати, щоб обманути і маршрутизатор, перехоплюючи його трафік.

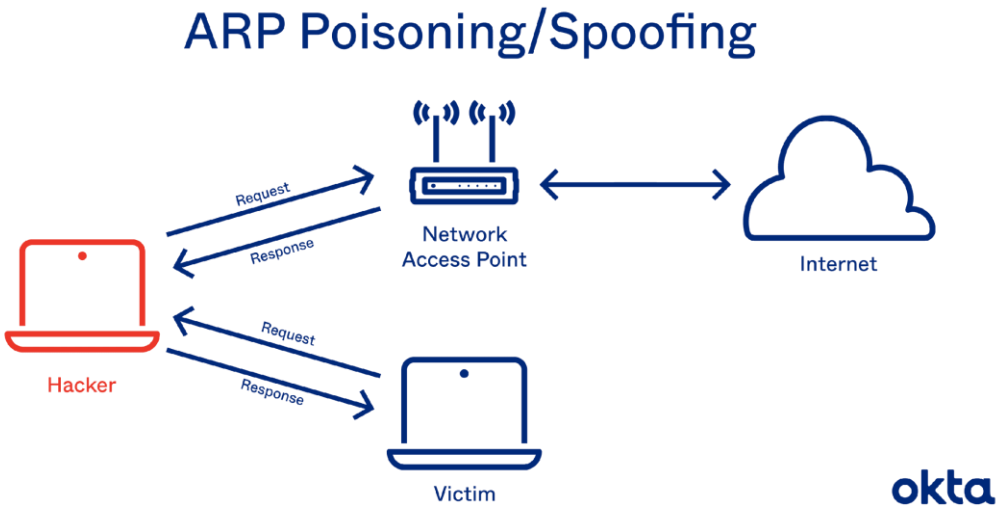

Рисунок 2.4 – Схема ARP-Spoofing атаки

Виявлення ARP-Spoofing атаки є складним завданням. Однією з ознак її проведення є те, що зловмисник наприкінці повертає MAC-таблицю в початковий стан, без чого мережа просто перестане працювати. Запити на зміну, а потім

відновлення таблиці можуть послужити індикатором для сповіщення про можливу атаку.

DoS атака - це атака, спрямована на вимкнення комп'ютера або мережі, що робить їх недоступними для запланованих користувачів. DoS-атаки досягають цього, переповнюючи ціль трафіком або надсилаючи їй інформацію, яка спричиняє збій.

В обох випадках DoS-атака позбавляє законних користувачів (тобто співробітників, членів або власників облікових записів) сервісу або ресурсу, на який вони розраховували. Жертвами DoS-атак часто стають веб-сервери відомих організацій, таких як банківські, комерційні та медіа-компанії, а також урядові та торгові організації. Хоча DoS-атаки зазвичай не призводять до крадіжки або втрати важливої інформації чи інших активів, вони можуть коштувати жертві багато часу та грошей на їх усунення [15].

Один з основних методів DoS-атак - атака типу «флуд». Вона відбуваються, коли система отримує занадто багато трафіку, який сервер не в змозі обробити, що призводить до сповільнення і, зрештою, зупинки роботи.

У випадку з SYN-flood таке виснаження досягається за допомогою постійного переривання процесу тристороннього рукостискання TCP (TCP Handshake), що призначений для встановлення з'єднання.

Під час рукостискання комп'ютери надсилають один одному наступні пакети (рисунок 2.5):

1. SYN: Комп'ютер-ініціатор (або активний клієнт) надсилає пакет з прапорцем SYN комп'ютеру-одержувачу (зазвичай серверу) зі значенням довільного числа (наприклад, 100), щоб дізнатися, чи доступні відкриті з'єднання;
2. SYN-ACK: Якщо комп'ютер-отримувач (також відомий як пасивний клієнт) має відкриті порти, які можуть прийняти з'єднання, він надсилає назад пакет підтвердження синхронізації (SYN-ACK)

комп'ютеру-ініціатору. Пакет містить два числа: власний номер SYN комп'ютера-отримувача, який може бути будь-яким довільним числом (наприклад, 200), і номер ACK, який дорівнює номеру SYN комп'ютера-ініціатора плюс один (наприклад, 101);

3. ACK: Комп'ютер-ініціатор (активний клієнт) надсилає пакет з прапорцем ACK назад до комп'ютера-отримувача, підтверджуючи отримання пакета SYN-ACK. Значення пакета дорівнює SYN комп'ютера-одержувача (надісланому на другому кроці) плюс одиниця (наприклад, 201). На цьому останньому кроці з'єднання встановлюється, і можна починати передачу даних [16].

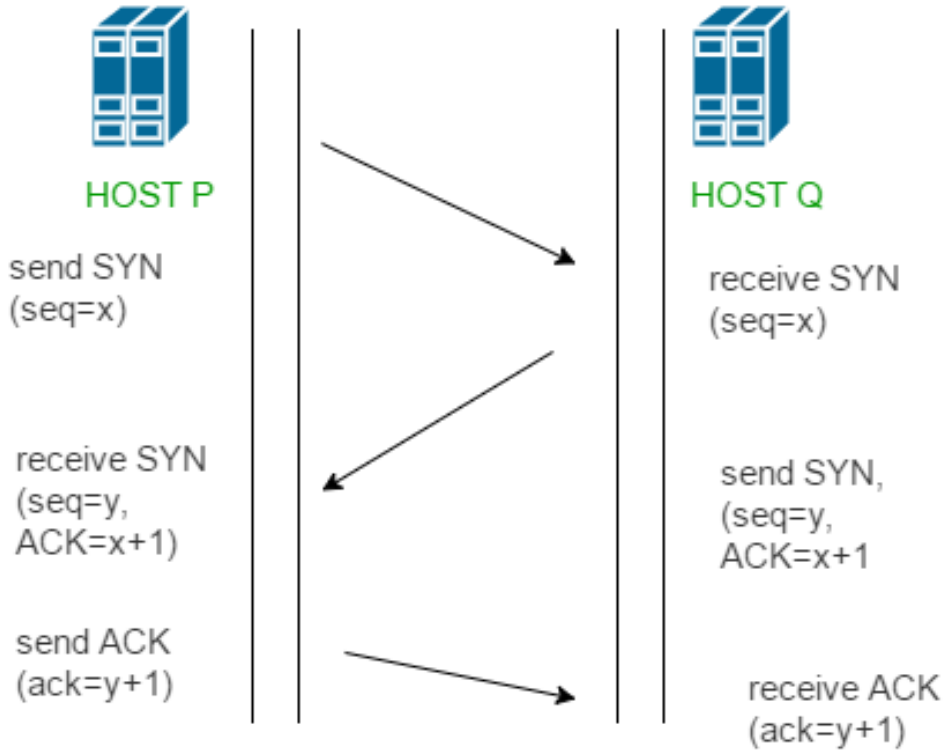

Рисунок 2.5 – Схема TCP рукостискання

Під час проведення атаки SYN-flood зловмисник постійно надсилає безліч SYN-пакетів системі жертви з підроблених IP-адрес, змушуючи атаковану систему надсилати SYN-ACK пакети неіснуючим комп'ютерам, водночас змушуючи систему відкривати свої порти та очікувати завершення таких рукостискань за допомогою пакетів ACK (рисунок 2.5.). Завдання зловмисника полягає в тому, щоб підтримувати чергу з'єднань заповненою таким чином, щоб не допустити нових

підключень. Через це клієнти, які не є зловмисниками, не можуть встановити зв'язок, або встановлюють його з істотними затримками.

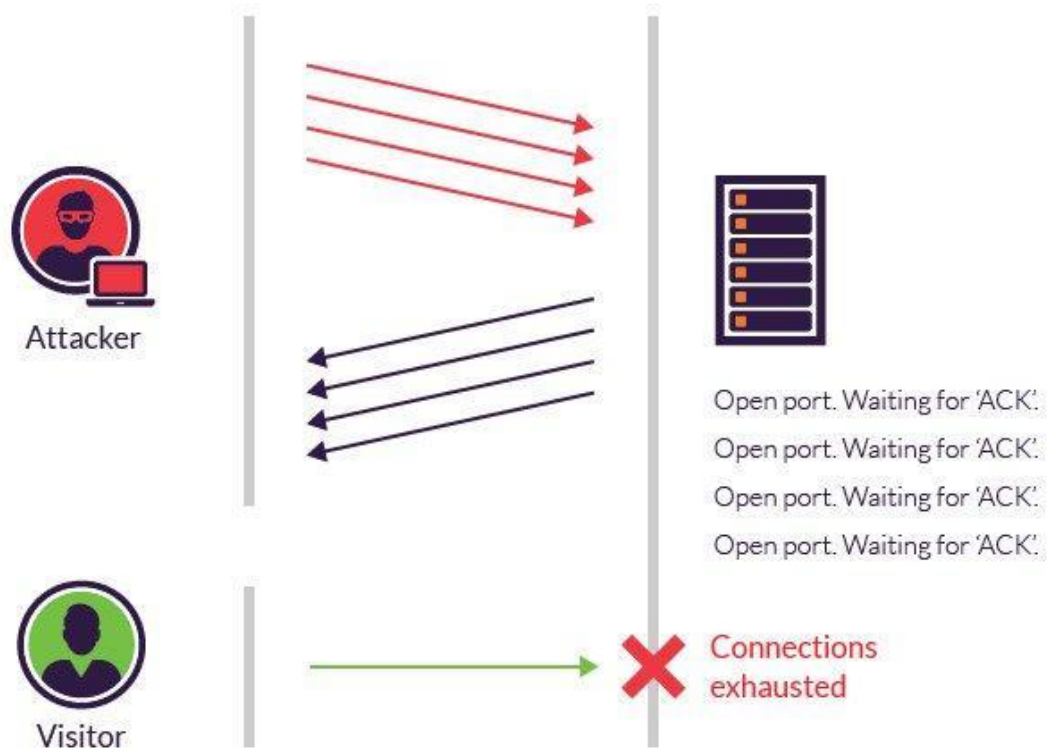

Рисунок 2.6 – Схема SYN-flood атаки

Головною ознакою SYN-flood атаки є величезна кількість TCP-пакетів з прапорцем SYN, які надсилаються на одну й ту саму адресу. Це явний індикатор можливого втручання в систему.

Крім DoS атак, процес TCP рукостискання експлуатують для розвідки мережі, а саме для сканування портів.

Сканування портів - це метод визначення того, які порти в мережі відкриті і можуть приймати або відправляти дані. Це також процес надсилання пакетів на певні порти пристрою та аналізу відповідей для виявлення вразливостей.

Мета сканування портів і мережі - визначити організацію IP-адрес, вузлів мережі і портів, щоб правильно визначити відкриті або вразливі місця розташування серверів і діагностувати рівень безпеки. Сканування мережі та портів може виявити наявність заходів безпеки, таких як мережевий екран між сервером і пристроєм користувача [17].

Шкідливе програмне забезпечення постійно сканує порти атакованої мережі. Так, кілька років тому ботнет Hajime ефективно запускав агресивне сканування порту 8291, щоб виявити загальнодоступні пристрої та експлуатувати інші підключені до нього пристрої.

Хробак запускає дуже агресивне SYN-сканування порту 8291, і якщо порт 8291 відкритий, він перевіряє інші поширені порти (80,81,82,8080,8081,8082,8089,8181,8880). Він також перевіряє версію пристрою і надсилає командний код експлойту. Така поведінка досить популярна серед наприклад троянів, тому виявлення можливого сканування портів також має бути серед завдань мережевих адміністраторів [18].

TCP сканування – як і SYN-flood атака, цей тип сканування використовує тристороннє рукостискання TCP щоб дізнатися, чи відкритий порт пристрою. Зловмисник починає TCP-з'єднання з надсилання SYN-пакету, а також вказує, до якого порту призначення він хоче підключитися.

Після цього, існує два можливих варіанти (рисунок 2.7):

1. Відкритий порт: Якщо цей порт пристрою відкритий, він відповість пакетом, в якому будуть увімкнені прапори SYN та ACK. Нарешті, зловмисник підвередить з’єднання пакетом з прапорцем AC, щоб потім скинути його за допомогою пакета з прапорцями RST та ACK;
2. Закритий порт: Якщо порт закрито, пристрій відповість на запит відповіддю, в якій будуть встановлені прапорці RST, ACK [19].

Таким чином, індикатором TCP сканування є той факт, що кожного разу, коли воно виконується, незалежно чи є успішним або неуспішним, одна з двох кінцевих точок повертає пакет з увімкненими двома прапорцями RST та ACK. Таким чином, безліч таких пакетів, відправлених на різні порти, може свідчити про те, що відбувається сканування портів.

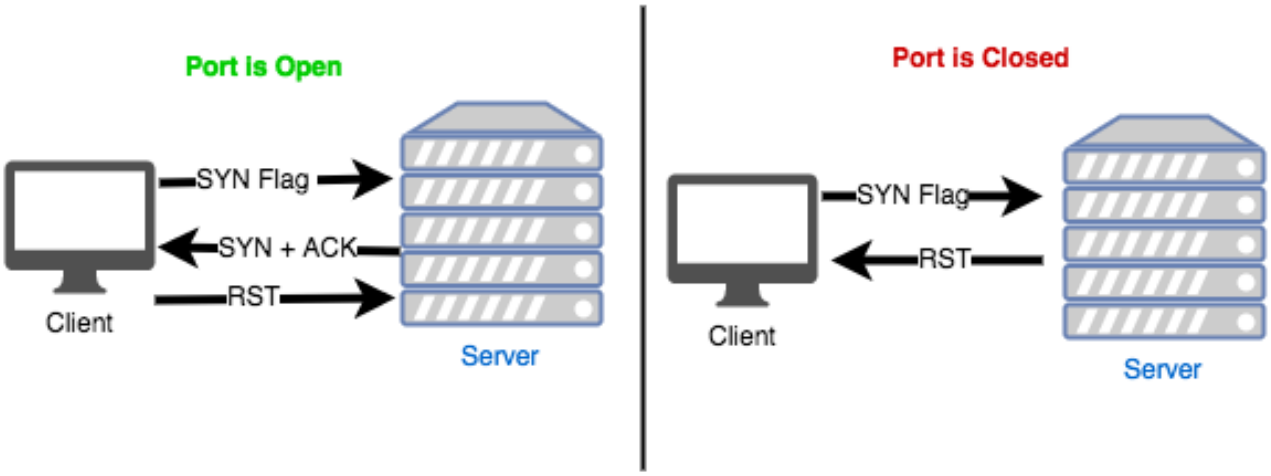

Рисунок 2.7 – Можливі відповіді при TCP скануванні

3 ПОБУДОВА ТА ТЕСТУВАННЯ СИСТЕМИ ДЛЯ АНАЛІЗУ МЕРЕЖЕВОГО ТРАФІКУ

3.1 Інтерфейс системи для аналізу мережевого трафіку

Веб-фреймворк Django для створення веб-додатків використовує архітектурний підхід MVT (Модель-представлення-шаблон). Модель, представлення та шаблон - це три шари з додатковими URL-адресами, кожен з яких має свою функцію і може використовуватися незалежно.

Модель - це "єдине авторитетне джерело знань про ваші дані", згідно з документацією Django. Вона містить ключові поля та поведінку даних, що зберігаються. Кожна модель зазвичай відповідає одній таблиці бази даних. Модель не знає інших шарів Django, тому що це простий клас Python. Інтерфейс (API) прикладного програмування є єдиним засобом зв'язку між шарами. Крім того, моделі зберігають елементи, пов'язані з маніпулюванням даними, такі як атрибути, користувацькі методи та бізнес-логіку. Об'єкти (набори даних) в оригінальній базі даних також можна створювати, читати, оновлювати та видаляти за допомогою моделей.

Представлення виконує деревоподібні завдання, включаючи прийом HTTP-запитів, реалізацію бізнес-логіки за допомогою класів і методів Python і генерацію HTTP-відповідей на клієнтські запити. Подання отримує дані з моделі і або надає доступ до конкретних даних для кожного шаблону, або готує дані для відображення.

Django містить ефективний механізм шаблонів і багатий інструментарій для власної мови розмітки. Шаблони - це HTML-файли, які використовуються для відображення даних. Вміст цих файлів може бути статичним або динамічним. Шаблон існує лише для представлення даних; він не містить бізнес-логіки.

URL-адреса визначає маршрутизацію та відображення URL-адрес у додатку. Вони визначені у файлі urls.py, який зіставляє запит користувача з конкретним представленням на основі шаблону URL-адреси.

Таким чином, модель відповідає за дані та бізнес-логіку, подання - за логіку представлення, шаблон - за користувацький інтерфейс, а URL-адреса - за маршрутизацію та відображення веб-додатку. Це призводить до кращої організації коду, зручності супроводу та масштабованості (рисунок 3.1) [20].

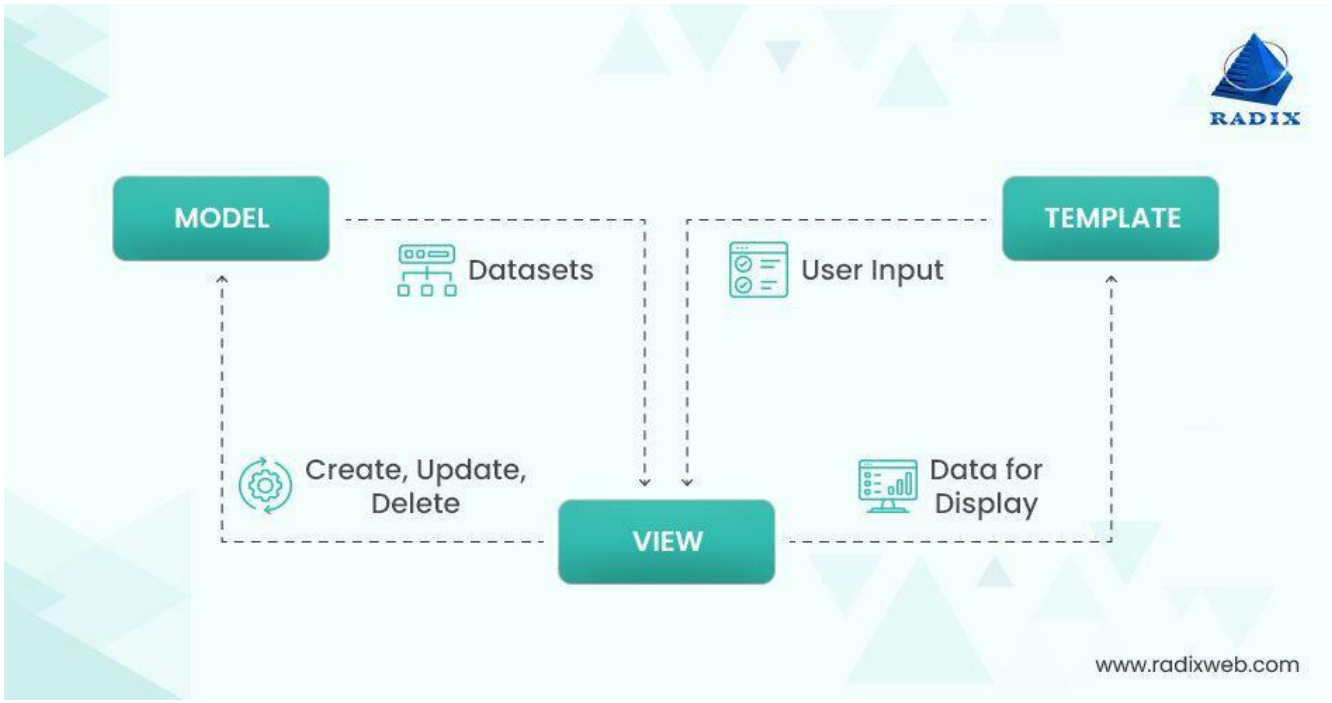

Рисунок 3.1 – Архітектура MVT

Для початку, потрібно створити папку проекта, створити в ній віртуальне середовище, та за допомогою пакетного менеджера pip встановити веб-фреймворк Django. Після чого, з'явиться можливість створити проект Django за допомогою команди:

```
django-admin startproject diploma
```

Це створює проект вже з готовою структурою:

```
diploma/
├── manage.py
├── diploma/
│   ├── __init__.py
│   ├── asgi.py
│   ├ settings.py
│   ├ urls.py
│   ├── wsgi.py
```

manage.py - цей скрипт слугує шлюзом до різних команд керування Django. Це інструмент, за допомогою ініціюється сервер розробки, створюються додатки, запускаються міграції тощо

Дочірній каталог diploma - це ядро, з якого походять всі функціональні можливості, формуючи веб-проект. У цьому каталозі знаходяться декілька ключових файлів:

- settings.py: цей файл містить налаштування, які конфігурують проект Django. Тут визначається, як функціонуватиме додаток: від конфігурації бази даних до списків проміжного програмного забезпечення;
- urls.py: цей файл визначає, яке представлення буде відображатися при доступі до певної URL-адреси.
- wsgi.py: скорочення від Web Server Gateway Interface, цей файл слугує точкою входу для додатку при розгортанні на продуктивному сервері. Це міст, що з'єднує додаток з веб-сервером, дозволяючи йому обробляти вхідні запити.
- asgi.py: подібно до wsgi.py, цей файл є точкою входу для асинхронних веб-серверів. Він розшифровується як Asynchronous Server Gateway Interface і полегшує обробку асинхронних HTTP-запитів.
- __init__.py: цей файл необхідний для організації та імпорту модулів у проекті [21].

Після створення директорії проекту, потрібно створити каталог для додатку, що робиться командою:

```
python manage.py startapp attack_detection
```

Головна різниця між проектом і додатком поягає в тому, що проект являє собою весь веб-додаток - набір налаштувань, конфігурацій і додатків, які працюють разом, утворюючи єдине ціле. Додаток, з іншого боку, є меншим, автономним

модулем у проекті, який слугує певній меті. Це може бути система автентифікації користувачів, блог або будь-яка інша автономна функціональність.

Каталог кожного додатку заповнений файлами, які разом визначають його поведінку та функціональність:

- models.py: це файл, де визначається моделі, використовуючи ORM (об'єктно-реляційне відображення) Django. Кожен клас моделі представляє таблицю в базі даних;
- views.py: цей файл інкапсулює логіку, яка визначає, як додаток взаємодіє із запитами користувачів;
- tests.py: цей файл містить модульні тести, щоб переконатися, що компоненти додатку функціонують належним чином;
- admin.py: цей файл налаштовує те, як моделі додатку будуть представлені в інтерфейсі адміністратора Django. Цей файл дозволяє адміністраторам легко керувати даними;
- migrations: цей каталог містить плани всіх змін у моделях вашого додатку;
- urls.py: цей файл зіставляє URL-адреси з поданнями;
- apps.py: цей файл керує конфігураціями конкретних додатків [21];.

Таким чином, архітектура проекта виглядає наступним чином:

```
diploma/
├── manage.py
├── diploma/
│   ├── __init__.py
│   ├── asgi.py
│   ├ settings.py
│   ├ urls.py
│   ├── wsgi.py
├── attack_detection/
│   ├── migrations/
│   │   └── __init__.py
│   ├── __init__.py
│   ├── admin.py
│   ├── apps.py
```

```
├── models.py
├── tests.py
├── urls.py
└── views.py
```

Для зв'язування проєкту і додатка потрібно додати додаток до списку додатків файлу settings.py:

```
INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'attack_detection.apps.AttackDetectionConfig',
]
```

Для збереження в базу даних можливих інцидентів потрібно створити для них модель. У файлі models.py було створено наступну модель:

```
class Incident(models.Model):
    time = models.DateTimeField("Час та дата
інциденту")
    source_IP = models.CharField("IP джерела",
max_length=20)
    source_port = models.CharField("Порт джерела",
max_length=7)
    destination_IP = models.CharField("IP
призначення", max_length=20)
    destination_port = models.CharField("Порт
призначення", max_length=7)
    incident_type = models.CharField("Тип інциденту",
max_length=25)
```

Тепер потрібно запустити міграцію, щоб таблиця інцидентів, разом з іншими стандартними таблицями Django, створилася в базі даних. Django спочатку створює базу даних sqlite3, але це можна змінити у файлі settings.py. Було використано стандартну базу даних. Для створення та запуску міграцій потрібні наступні команди:

```
python manage.py makemigrations
python manage.py migrate
```

Для виведення даних у панель адміністатора потрібно зареєструвати створену модель у файлі admin.py:

```
class IncidentAdmin(admin.ModelAdmin):
    list_display = ["time", "source_IP",
"destination_IP", "destination_port", "incident_type"]
admin.site.register(Incident, IncidentAdmin)
```

Також потрібно створити акаунт адміністратора, що робиться командою:

```
python manage.py createsuperuser
```

3.2 Система для аналізу мережевого трафіку

Для створення системи для аналізу мережевого трафіку потрібно ще додатково кілька файлів і каталогів:

- Файл, який має в собі функції для виявлення можливих інцидентів;
- Каталог для зберігання файлів із мережевим трафіком;
- Файл, який виявляє нові файли з мережевим трафіком у каталозі та викликає функцію для виявлення інцидентів.

Було створено файл із назвою alert_functions.py. У ньому містяться функції виявлення атак, засновані на індикаторах інцидентів, які були створені в другому розділі. Як аргумент, функції приймають мережевий пакет, і як результат або надсилають значення False, якщо немає ознак інідента, або створюють об'єкт у базі даних і надсилають значення True, якщо ознаку виявлено.

Для виявлення TCP сканування розроблено такий алгоритм:

1. Вводиться словник adresses, який зберігає значення "ip адреса": кількість пакетів із прапором RST, ACK, а також список Port_Number, який зберігає список портів, які сканують;

2. Функція перевіряє, чи є в пакеті протоколи TCP і IP: якщо є, то записує у змінні прапори пакета, порт призначення та адресу призначення; Якщо таких протоколів немає в пакеті, функція повертає значення False;
3. Відбувається перевірка, чи є збережена адреса в словнику adresses, і якщо адреси немає, то створити пару з цією адресою;
4. Функція перевіряє, чи є прапор RST,ACK (еквівалент 0x014) у пакеті, і якщо такого прапора немає, функція повертає False;
5. Якщо такий прапор є в пакеті, то в список Port_Number додається порт, а також для адреси в словнику adresses значення збільшується на 1. Після чого відбувається перевірка, чи більша за 20 довжина Port_Number і чи більше за 20 значення адреси в adresses;
6. Якщо більше, то з пакета беруться необхідні дані, які зберігаються в базу даних.

Для виявлення SYN Flood атаки розроблено такий алгоритм:

1. Вводиться словник SYN_Table, у якому зберігаються пари "ip адреса": кількість SYN пакетів;
2. Функція перевіряє, чи є протоколи TCP і IP, і або зберігає потрібні значення у змінні, або повертає False;
3. Функція перевіряє, чи є адреса одержувача пакета в SYN_Table, і якщо в пакеті є прапор SYN, то збільшує значення цієї адреси в словнику на 1;
4. Функція перевіряє значення адреси в SYN_Table, якщо воно більше 10000, то записує в базу даних значення пакета.

Для виявлення ARP Spoofing атаки розроблено такий алгоритм:

1. Вводиться словник IP_MAC_Table, в якому зберігаються пари "MAC-адреса": "IP-адреса";
2. Функція перевіряє наявність протоколів TCP і IP;

3. Функція перевіряє наявність протоколів TCP і IP;
4. Функція перевіряє, чи є MAC-адреса одержувача пакета в IP_MAC_Table, а також перевіряє, чи відповідає IP-адреса MAC-адресі пакета;
5. Якщо не відповідає, то функція записує дані пакета в базу даних.

Також була прописана функція alerting(), яка за циклом перевіряє пакети у файлі за допомогою інших функцій та зупиняє цикл, якщо одна з функцій знаходить інцидент (повертає True).

Було створено каталог PCAPs, який призначений для зберігання файлів із мережевим трафіком. Також було створено файл detector.py, який перевіряє появу нових файлів у каталозі PCAPs. Це було досягнуто за допомогою бібілотеки watchdog.

В detector.py було прописано клас MyHandler, який зберігає в собі назву нового файл PCAPs, після чого, за допомогою MyHandler та вбудованого класу Observer був написаний нескінченний цикл, який реагує на появу нового файлу всередині PCAPs та викликає функцію alerting().

Після створення каталогу та файлів схема проекту набула такого вигляду:

Код цих файлів відображено у Додатку Б

```
diploma/
├── manage.py
├── alerting.py
├── detector.py
├── PCAPs/
├── diploma/
│   ├── __init__.py
│   ├── asgi.py
│   ├ settings.py
│   ├ urls.py
│   ├── wsgi.py
├── attack_detection/
    ├── migrations/
    │   └── __init__.py
```

```
├── __init__.py
├── admin.py
├── apps.py
├── models.py
├── tests.py
├── urls.py
└── views.py
```

3.3 Тестування системи для аналізу мережевого трафіку

Для тестування системи для аналізу мережевого трафіку було вирішено згенерувати файли на основі моделювання дій зловмисника. Так як будь-яке шкідливе втручання в інші комп’ютери та системи є незаконною та неетичною діяльністю, було побудовано віртуальну лабораторію для цих цілей [3].

Після налаштування лабораторії можна переходити до моделювання атак.

Атака типу ARP-Spoofing може бути реалізована за допомогою набору інструментів dsniff. Після його встановлення з'явиться можливість використовувати інструмент arpspoof. Наступна команда створить кілька пакетів, в яких буде вказано, що MAC-адреса Kali Purple відповідає IP-адресі pfSense (рисунок 3.2.):

```
arpspoof -i eth0 -t 192.168.1.102 192.168.1.1
```

```
└─# arpspoof -i eth0 -t 192.168.1.102 192.168.1.1
8:0:27:8b:4e:28 8:0:27:58:17:8a 0806 42: arp reply 192.168.1.1 is-at 8:0:27:8b:4e:28
8:0:27:8b:4e:28 8:0:27:58:17:8a 0806 42: arp reply 192.168.1.1 is-at 8:0:27:8b:4e:28
8:0:27:8b:4e:28 8:0:27:58:17:8a 0806 42: arp reply 192.168.1.1 is-at 8:0:27:8b:4e:28
8:0:27:8b:4e:28 8:0:27:58:17:8a 0806 42: arp reply 192.168.1.1 is-at 8:0:27:8b:4e:28
8:0:27:8b:4e:28 8:0:27:58:17:8a 0806 42: arp reply 192.168.1.1 is-at 8:0:27:8b:4e:28
8:0:27:8b:4e:28 8:0:27:58:17:8a 0806 42: arp reply 192.168.1.1 is-at 8:0:27:8b:4e:28
8:0:27:8b:4e:28 8:0:27:58:17:8a 0806 42: arp reply 192.168.1.1 is-at 8:0:27:8b:4e:28
8:0:27:8b:4e:28 8:0:27:58:17:8a 0806 42: arp reply 192.168.1.1 is-at 8:0:27:8b:4e:28
8:0:27:8b:4e:28 8:0:27:58:17:8a 0806 42: arp reply 192.168.1.1 is-at 8:0:27:8b:4e:28
8:0:27:8b:4e:28 8:0:27:58:17:8a 0806 42: arp reply 192.168.1.1 is-at 8:0:27:8b:4e:28
```

Рисунок 3.2 – Процес роботи arpspoof

У результаті захоплення трафіку під час роботи цього інструменту було отримано файл ARPSpoofing.pcapng.

Для проведення такого виду атак було використано інструмент hping3, який дає змогу випадково згенерувати IP-адреси атакуючого. Наступна команда відправила 5000 SYN пакетів віртуальній машині Metasploitable (рис. 3.3.):

```
hping3 -c 5000 -d 120 -S -w 32 -p 80 --flood --rand-source 192.168.1.102
```

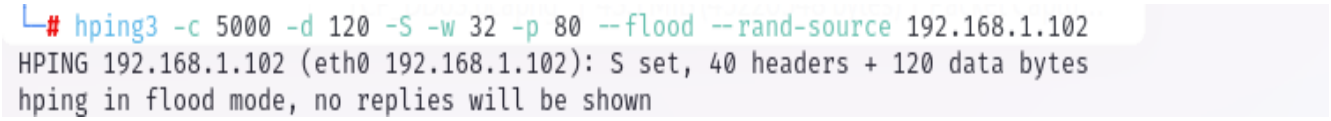

Рисунок 3.3 – Процес роботи hping3

У результаті захоплення трафіку під час роботи цього інструменту було отримано файл TCP_DDOs.pcapng

Для проведення TCP сканування ідеально підійде інструмент Nmap. Наступна команда просканує Metasploitable на наявність відкритих портів за допомогою TCP сканування (рисунок 3.4):

```
nmap -sT 192.168.1.102
```

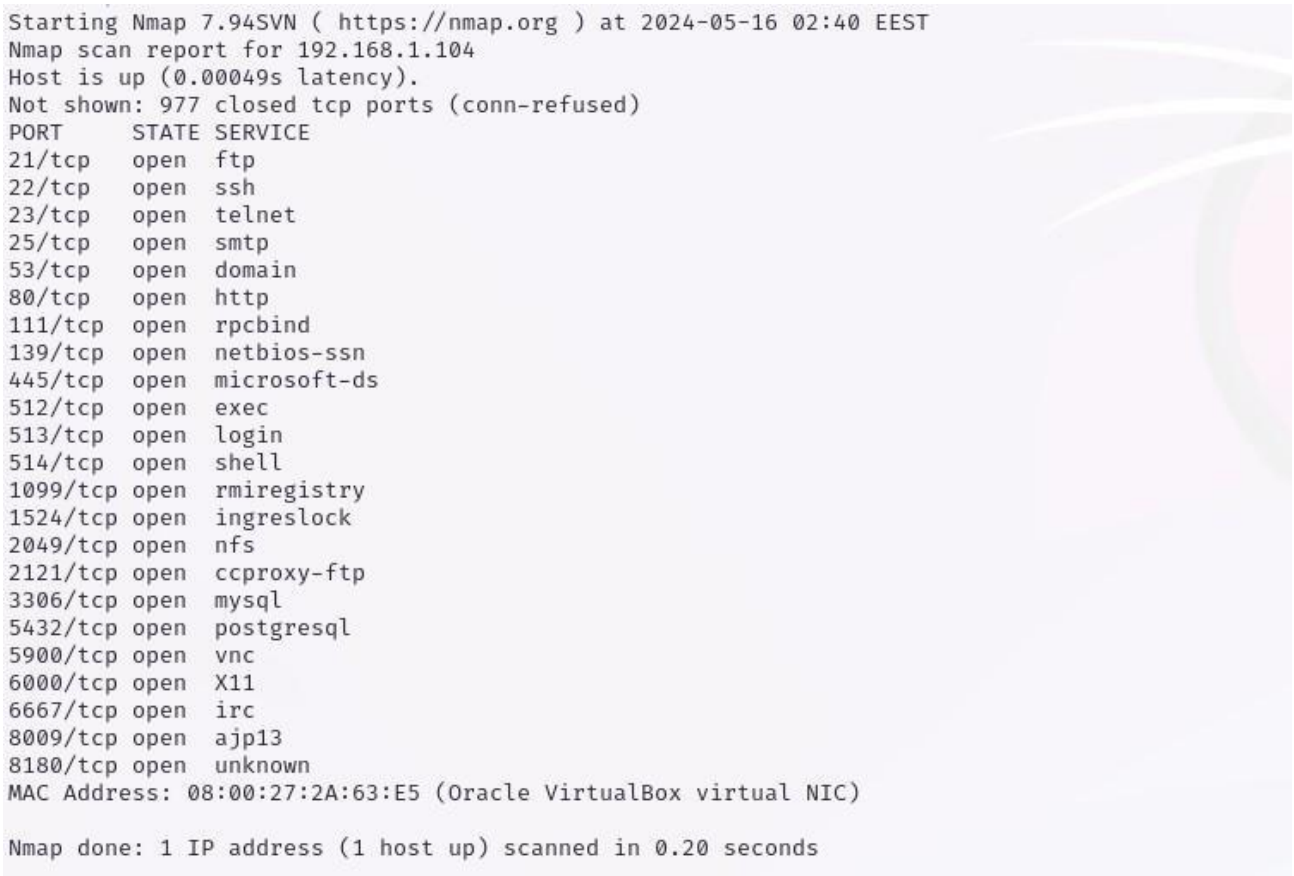

Рисунок 3.4 – Процес роботи Nmap

У результаті захоплення трафіку під час роботи цього інструменту було отримано файл TCP_Scan.pcapng

Таким чином було отримано два файли, за допомогою яких буде можливість оцінити роботу розробленої системи.

Після запуску файлу detector.py та копіювання в результаті тестування файлів у каталог PCAPs були отримані наступні результати, які можна переглянути через панель адміністратора Django (рисунок 3.4):

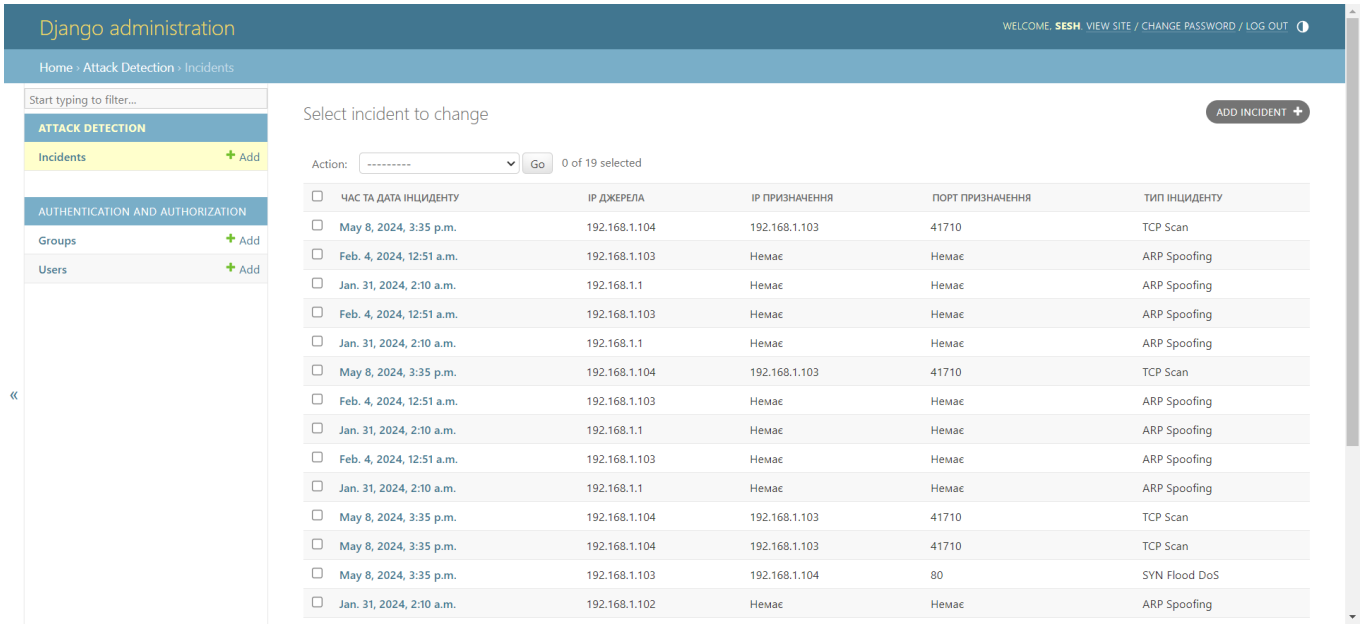

Рисунок 3.5 – Результати тестування

ВИСНОВОК

В роботі розглянуте завдання проєктування, реалізації та тестування системи для аналізу мережового трафіку на наявність в ньому шкідливої активності.

У процесі досягнення мети дослідження проведено аналіз існуючих інтрументів виявлення інцидентів кібербезпеки та найчастіших кібератак. На основі результатів аналізу визначено функціональні вимоги до системи, вибрано засоби розробки програмного застосунку.

У рамках програмної реалізації системи виконані такі роботи:

1. Розроблені механізми автоматичного аналізу мережевих пакетів на наявність можливих загроз;
2. До системи додано інтерфейс для перегляду інцидентів за допомогою веб-фреймворку Django для перегляду отриманих реузультатів;
3. Проведено серію симуляцій дій зловмисника з метою протестування створеної системи.

З метою подальших покращень системи можливе додавання додаткових можливостей для аналізу мережевого трафіку, створення нових функцій для виявлення інших інцидентів, а також розширення функціональності самої системи, наприклад додавання графіків для візуалізації інцидентів.

Таким чином, таку систему можна буде використовувати для невеликих і середніх мереж, наприклад для університетської мережі.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Кількість кібератак у 2023 році зросла на 16%. URL: https://www.epravda.com.ua/news/2024/01/31/709355/
2. Дрига А. ВИКОРИСТАННЯ БІБЛІОТЕКИ SCAPY ДЛЯ АНАЛІЗУ МЕРЕЖЕВОГО ТРАФІКУ. Матеріали Міжнародної науково-практично конференці «Кіберпростір в умовах війни та глобальних викликів XXI століття: теорія та практика» (м. Одеса, 24 листопада 2023 р.). Одеса, 2023. с.102-104
3. Дрига А. СТВОРЕННЯ ВІРТУЛЬНОЇ ЛАБОРАТОРІЇ ДЛЯ МОДЕЛЮВАННЯ ПОТЕНЦІЙНИХ АТАК. Інформаційне суспільство: проблеми та перспективи : матеріали IX Всеукр. наук.-практич. конф. (м. Одеса, 24 травня 2024 р.) / відп. ред. Н.І.Логінова. Одеса,2024, ДЕПОНОВАНО
4. Вірус Petya - лише один з можливих векторів кібератак. URL: https://www2.deloitte.com/ua/uk/pages/press-room/expert/2017/petya-virus-one-of-possible-cyber-attack.html
5. incident - Glossary - NIST Computer Security Resource Center. URL: https://csrc.nist.gov/glossary/term/incident
6. What is incident response?. URL: https://www.atlassian.com/incident-management/incident-response#:~:text=Incident%20response%20refers%20to%20the,by%20cyber%20threats%20or%20breaches
7. Computer Security Incident Handling Guide. URL: https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-61r2.pdf
8. Intrusion Detection System (IDS). URL: https://www.geeksforgeeks.org/intrusion-detection-system-ids/
9. Intrusion Prevention System (IPS). URL: https://www.geeksforgeeks.org/intrusion-prevention-system-ips/

10. What Is Security Information and Event Management (SIEM)? URL: https://www.paloaltonetworks.com/cyberpedia/what-is-security-information-and-event-management-SIEM
11. Matplotlib — Visualization with Python. URL: https://matplotlib.org/
12. 2023 in Review: DDoS Attacks Report by StormWall. URL: https://stormwall.network/ddos-attack-report-2023
13. Man-in-the-Middle (MitM) attacks reaching inboxes increase 35% since 2022. URL: https://securityboulevard.com/2023/05/man-in-the-middle-mitm-attacks-reaching-inboxes-increase-35-since-2022/
14. Man in the middle (MITM) attack. URL: https://www.imperva.com/learn/application-security/man-in-the-middle-attack-mitm/
15. What is a denial of service attack (DoS)? URL: https://www.paloaltonetworks.com/cyberpedia/what-is-a-denial-of-service-attack-dos
16. What Is a Three-Way Handshake? URL: https://www.coursera.org/articles/three-way-handshake
17. What is port scanning? URL: https://www.avast.com/en-us/business/resources/what-is-port-scanning#pc
18. Aggressive Scans by Hajime Botnet Targeting Port 8291 With a new Exploit. URL: https://gbhackers.com/hajime-botnet-port-8291/
19. Nmap Detection with Wireshark. URL: https://medium.com/@thismanera/nmap-detection-with-wireshark-e780c73a0823
20. Django Web App Development Framework: A Detailed Guide 2024. URL: https://radixweb.com/blog/what-is-django
21. Django Project Structure: A Comprehensive Guide. URL: https://medium.com/django-unleashed/django-project-structure-a-comprehensive-guide-4b2ddbf2b6b8

Додаток А

```
from scapy.all import rdpcap, PcapReader, sniff

from pyshark import FileCapture

import matplotlib.pyplot as plt

import time

first_pcap_file = 'TCP_DDOs.pcapng'

second_pcap_file = 'ARPSpoofing.pcapng'

pyshark_results = []

scapy_first_results = []

scapy_second_results = []

scapy_third_results = []

def plot():

    plt.hist([pyshark_results, scapy_first_results,
scapy_second_results, scapy_third_results],

             color=['red', 'tan', 'lime', 'green'],

             label=['PyShark', 'Scapy (rdpcap)', 'Scapy
(PcapReader)', 'Scapy (sniff)'],

             alpha=0.7)

    plt.legend()

    plt.xlabel('Execution Time (seconds)')

    plt.ylabel('Frequency')

    plt.title('Execution Time Comparison')
```

```
    plt.show()
def pyshark():
    for i in range(10):
        start_time = time.perf_counter()
        file = FileCapture(second_pcap_file)
        print(file[0])
        finish_time = time.perf_counter()
        pyshark_results.append(finish_time - start_time)
        print(f'Час виконання {finish_time - start_time}')
def scapy_first():
    for i in range(10):
        start_time = time.perf_counter()
        file = rdpcap(second_pcap_file)
        print(file[0])
        finish_time = time.perf_counter()
        scapy_first_results.append(finish_time -
start_time)
        print(f'Час виконання {finish_time - start_time}')
def scapy_second():
    for i in range(10):
        start_time = time.perf_counter()
        file = PcapReader(second_pcap_file)
```

```
        print(file[0])
        finish_time = time.perf_counter()
        scapy_second_results.append(finish_time -
start_time)
        print(f'Час виконання {finish_time - start_time}')
def scapy_third():
    for i in range(10):
        start_time = time.perf_counter()
        file = sniff(offline=second_pcap_file, store=0)
        print(file[0])
        finish_time = time.perf_counter()
        scapy_third_results.append(finish_time -
start_time)
        print(f'Час виконання {finish_time - start_time}')
if __name__ == '__main__':
    scapy_first()
    pyshark()
    scapy_second()
    scapy_third()
    plot()
```

Додаток Б

Файл alert_functions.py

```
import datetime
from scapy.all import PcapReader
import sys, os, django
sys.path.append(r"C:\Users\driga\OneDrive\Desktop\diploma")
os.environ.setdefault("DJANGO_SETTINGS_MODULE",
"diploma.settings")
django.setup()
from attack_detection.models import Incident
MAX_SYN_PACKETS = 10000
IP_MAC_Table = {}
SYN_Table = {}
Port_Number = []
adresses = {}
def TCPScanDetect(packet):
    if packet.haslayer('TCP') and packet.haslayer('IP'):
        flags = packet['TCP'].flags
        destination_port = packet['TCP'].dport
        destination_IP = packet['IP'].dst
    else:
        return False
```

```
    if destination_IP not in adresses.keys():
        adresses[destination_IP] = 0
    if flags == 0x014:
        Port_Number.append(destination_port)
        adresses[destination_IP] += 1
        if len(Port_Number) > 20 and
adresses[destination_IP] > 20:
            time =
datetime.datetime.fromtimestamp(packet.time)
            source_IP = packet['IP'].src
            source_port = packet['IP'].sport
            incident = Incident(time=time,
source_IP=source_IP, source_port=source_port,
destination_IP=destination_IP,

destination_port=destination_port, incident_type="TCP
Scan")
            incident.save()
            return True
    else:
        return False
def SYNFloodDetect(packet):
    if packet.haslayer('TCP') and packet.haslayer('IP'):
```

```
        flags = packet['TCP'].flags
        destination_IP = packet['IP'].dst
    else:
        return False
    if destination_IP not in SYN_Table.keys():
        SYN_Table[destination_IP] = 0
    if flags == 'S':
        SYN_Table[destination_IP] += 1
    for IP in SYN_Table.keys():
        if SYN_Table[IP] > 10000:
            time =
datetime.datetime.fromtimestamp(packet.time)
            destination_port = packet['TCP'].dport
            source_port = packet['TCP'].sport
            source_IP = packet['IP'].src
            incident = Incident(time=time,
source_IP=source_IP, source_port=source_port,
destination_IP=destination_IP,

destination_port=destination_port, incident_type="SYN Flood
DoS")
            incident.save()
            return True
```

```
        else:
            return False
def ARPSpoofingDetect(packet):
    if packet.haslayer('ARP') and packet.haslayer('Ether'):
        source_IP = packet['ARP'].psrc
        source_MAC = packet['Ether'].src
    else:
        return False
    if source_MAC in IP_MAC_Table.keys():
        if IP_MAC_Table[source_MAC] != source_IP:
            try:
                old_IP = IP_MAC_Table[source_MAC]
            except:
                old_IP = "Невідомий"
            time =
datetime.datetime.fromtimestamp(packet.time)
            if packet.haslayer('TCP'):
                destination_port = packet['TCP'].dport
                source_port = packet['TCP'].sport
            elif packet.haslayer('UPD'):
                destination_port = packet['UDP'].dport
                source_port = packet['UPD'].sport
```

```
            else:
                destination_port = "Немає"
                source_port = "Немає"
            if packet.haslayer('IP'):
                destination_IP = packet['IP'].dst
            else:
                destination_IP = "Немає"
            incident = Incident(time=time,
source_IP=source_IP, source_port=source_port,
destination_IP=destination_IP,

destination_port=destination_port, incident_type="ARP
Spoofing")
            incident.save()
            return True
        else:
            return False
    else:
        IP_MAC_Table[source_MAC] = source_IP
        return False
def alerting(path):
    file = PcapReader(path)
    is_incident = False
```

```
for packet in file:
    is_incident = TCPScanDetect(packet)
    if is_incident:
        print("Произошел ТСП СКАН")
        SYN_Table = {}
        Port_Number = []
        adresses = {}
        break
    is_incident = SYNFloodDetect(packet)
    if is_incident:
        print("Произошел ДДОС")
        SYN_Table = {}
        Port_Number = []
        adresses = {}
        break
    is_incident = ARPSpoofingDetect(packet)
    if is_incident:
        print("Произошел СПУФИНГ")
        SYN_Table = {}
        Port_Number = []
        adresses = {}
        break
```

Файл detector.py

```
import time
from watchdog.observers import Observer
from watchdog.events import FileSystemEventHandler
from alert_functions import alerting
class MyHandler(FileSystemEventHandler):
    def on_created(self, event):
        if event.is_directory:
            return
        time.sleep(3)
        alerting(event.src_path[2:])
if __name__ == "__main__":
    path = "./PCAPs"
    event_handler = MyHandler()
    observer = Observer()
    observer.schedule(event_handler, path, recursive=False)
    observer.start()
    try:
        while True:
            time.sleep(1)
    except KeyboardInterrupt:
        observer.stop()
```

```
    observer.join()
```

Файл models.py

```
from django.db import models

class Incident(models.Model):

    time = models.DateTimeField("Час та дата інциденту")

    source_IP = models.CharField("IP джерела",
max_length=20)

    source_port = models.CharField("Порт джерела",
max_length=7)

    destination_IP = models.CharField("IP призначення",
max_length=20)

    destination_port = models.CharField("Порт призначення",
max_length=7)

    incident_type = models.CharField("Тип інциденту",
max_length=25)
```

Файл admin.py

```
from django.contrib import admin

from .models import Incident

class IncidentAdmin(admin.ModelAdmin):

    list_display = ["time", "source_IP", "destination_IP",
"destination_port", "incident_type"]

admin.site.register(Incident, IncidentAdmin)
```