МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНЕ НЕКОМЕРЦІЙНЕ ПІДПРИЄМСТВО
«ДЕРЖАВНИЙ УНІВЕРСИТЕТ
«КИЇВСЬКИЙ АВІАЦІЙНИЙ ІНСТИТУТ»
ФАКУЛЬТЕТ КОМП'ЮТЕРНИХ НАУК ТА ТЕХНОЛОГІЙ
КАФЕДРА КІБЕРБЕЗПЕКИ

ДОПУСТИТИ ДО ЗАХИСТУ
Завідувач кафедри кібербезпеки

_____Анна ІЛЬЄНКО

«___» _____2024 р.

КВАЛІФІКАЦІЙНА РОБОТА

(ПОЯСНЮВАЛЬНА ЗАПИСКА)

ЗДОБУВАЧА ВИЩОЇ ОСВІТИ СТУПЕНЯ «МАГІСТР»

Тема: Програмний застосунок моніторингу мережевого трафіку з метою виявлення та блокування зовнішнього вторгнення в комп'ютерні системи

Виконавець: Андрій БОРЩ

Керівник: к.т.н. Олена ВИСОЦЬКА

Нормоконтролер: к.т.н., доцент Андрій ПЕТРЕНКО

Київ 2024

ДЕРЖАВНЕ НЕКОМЕРЦІЙНЕ ПІДПРИЄМСТВО
«ДЕРЖАВНИЙ УНІВЕРСИТЕТ
«КИЇВСЬКИЙ АВІАЦІЙНИЙ ІНСТИТУТ»

Факультет комп'ютерних наук та технологій
Кафедра кібербезпеки
Спеціальність 125 «Кібербезпека»
Освітньо-професійна програма «Безпека інформаційних і комунікаційних систем»
Форма навчання денна

ЗАТВЕРДЖУЮ
Завідувач кафедри кібербезпеки

_____Анна ІЛЬЄНКО
«___» _____2024 р.

ЗАВДАННЯ

на виконання кваліфікаційної роботи

Борща Андрія Андрійовича

1.Тема кваліфікаційної роботи: Програмний застосунок моніторингу мережевого трафіку з метою виявлення та блокування зовнішнього вторгнення в комп'ютерні системи

затверджена наказом ректора від « 30 » серпня 2024 року № 1695 /ст.

2. Термін виконання роботи: з 30.08.2024 по 15.12.2024

3. Вихідні дані до проєкту (роботи): проаналізувати методи моніторингу мережевого трафіку, на основі проведеного аналізу обрати найбільш ефективні методи виявлення і блокування вторгнення та розробити програмний застосунок виявлення та блокування зовнішнього вторгнення в комп'ютерні системи, провести тестування системи.

4. Зміст пояснювальної записки (перелік питань, що підлягають розробці): аналіз методів моніторингу мережевого трафіку в комп'ютерних системах; реалізація методів виявлення та блокування зовнішнього вторгнення в комп'ютерні системи; описання розробленого програмного застосунку виявлення та блокування зовнішнього вторгнення; тестування програмного застосунку виявлення та блокування зовнішнього вторгнення.

5. Перелік обов'язкового графічного матеріалу: презентація.

6. Календарний план-графік

№ п/п	Завдання	Термін виконання	Примітка
1	Провести аналіз літератури за темою кваліфікаційної роботи	30.08.24 - 05.09.24	*Виконано*
2	Провести аналіз методів моніторингу мережевого трафіку з метою виявлення та блокування зовнішнього вторгнення в комп'ютерні системи	06.09.24- 20.09.24	*Виконано*
3	Провести порівняльний аналіз виявлення зовнішнього вторгнення в комп'ютерні системи	21.09.24- 28.09.24	*Виконано*
4	Провести порівняльний аналіз існуючих методів блокування зовнішнього вторгнення в комп'ютерні системи	29.09.24- 07.10.24	*Виконано*
5	Розробити програмний застосунок моніторингу мережевого трафіку, який буде використовувати інструменти перехоплення трафіку і аналізувати мережеві пакети засобами ШІ з метою виявлення та блокування зовнішнього вторгнення	08.10.24- 21.10.24	*Виконано*
6	Провести тестування розробленого програмного застосунку на точність виявлення, рівень помилкових спрацьовувань, середній час обробки, що дасть змогу дослідити доцільність використання розробленого застосунку для вирішення поставленої задачі	26.11.24- 27.11.24	*Виконано*

7. Дата видачі завдання « 30» серпня 2024 р.

Керівник кваліфікаційної роботи_____ Олена ВИСОЦЬКА
(підпис керівника)

Завдання прийняв до виконання_____ Андрій БОРЩ
(підпис студента)

РЕФЕРАТ

Пояснювальна записка до кваліфікаційної роботи «Програмний застосунок моніторингу мережевого трафіку з метою виявлення та блокування зовнішнього вторгнення в комп'ютерні системи» складається з: 106 с., 13 рис., 3 таблиці, 31 літературне джерело, 1 додаток.

Об'єкт дослідження: процес моніторингу та аналізу мережевого трафіку в комп'ютерних системах з метою забезпечення їх захисту від зовнішніх загроз.

Предмет дослідження: методи моніторингу мережевого трафіку для виявлення і блокування вторгнень у комп'ютерні системи.

Мета кваліфікаційної роботи: розробити програмний застосунок моніторингу мережевого трафіку з метою виявлення та блокування зовнішнього вторгнення в комп'ютерні системи.

Методи дослідження: аналіз сучасних методів моніторингу мережевого трафіку та існуючих засобів захисту комп’ютерних систем для визначення поширених методів виявлення зовнішнього вторгнення в комп'ютерні системи, моделювання роботи системи для визначення ефективності прийнятих рішень для виявлення аномалій, експериментальні дослідження розробленого застосунку на реальних даних для оцінки його доцільності в умовах виявлення різних типів атак.

Наукова новизна роботи полягає у розробці програмного застосунку, який за рахунок аналізу вмісту пакетів забезпечує моніторинг мережевого трафіку в реальному часі та автоматично блокує зовнішні атаки. Такий підхід дозволяє не лише виявляти атаки, але й негайно реагувати на них, що є надзвичайно важливим для захисту критичних систем.

Практична цінність роботи полягає у створенні програмного застосунку для моніторингу мережевого трафіку, який завдяки інтеграції штучного інтелекту дозволяє виявляти та блокувати загрози без використання додаткових апаратних потужностей.

Результати кваліфікаційної роботи рекомендується використовувати:

1) у комерційних і державних організаціях для підвищення рівня інформаційної безпеки шляхом оперативного виявлення та блокування зовнішніх вторгнень;

2) в освітньому процесі у навчальних закладах, які готують спеціалістів у галузі кібербезпеки та інформаційних технологій, для ознайомлення з сучасними методами аналізу мережевого трафіку,

3) у розробці систем безпеки для критичної інфраструктури, де необхідно забезпечити високий рівень захисту від зовнішніх загроз.

МОНІТОРИНГ, МЕРЕЖА, ТРАФІК, АРХІТЕКТУРА ДОДАТКІВ, ОБРОБКА ДАНИХ, ЗАХИСТ, АНОМАЛІЇ, ГРАФІЧНИЙ ІНТЕРФЕЙС КОРИСТУВАЧА.

ЗМІСТ

API – *Application Programming Interface*

CGI – *Common Gateway Interface*

CRM – *Customer Relationship Management*

DNS – *Domain Name Service*

FTP – *File Transfer Protocol*

GUI – *Graphical User Interface*

LAN – *Local Area Network*

TCP/IP – *Transmission Control Protocol/Internet Protocol*

Розвиток інформаційних технологій та інтернет-інфраструктури створив нові можливості для покращення ефективності бізнес-процесів, державного управління, а також особистого життя кожної людини. Разом з тим, використання комп’ютерних мереж стало невід’ємною частиною сучасного суспільства, що створює нові виклики в забезпеченні безпеки комп'ютерних систем від зовнішніх загроз. Постійне зростання обсягів мережевого трафіку та збільшення кількості підключених до Інтернету пристроїв (особливо у контексті Інтернету речей, *IoT*) значно підвищили ризики несанкціонованого доступу, атак та інших загроз, спрямованих на комп'ютерні системи. Враховуючи ці тенденції, особливого значення набуває моніторинг мережевого трафіку та захист від зовнішніх загроз.

Однією з основних проблем сучасної кібербезпеки є необхідність оперативного виявлення і блокування вторгнень у комп'ютерні системи. Вторгнення можуть бути спрямовані на крадіжку даних, руйнування інформаційних систем, порушення роботи мереж та нанесення збитків компаніям і державним установам. Кібератаки мають різноманітні форми, зокрема *DDoS*-атаки, атаки типу "людина посередині" (*MITM*), шкідливе програмне забезпечення та інші типи загроз, які можуть бути втілені через мережевий трафік. Сучасні атаки стають дедалі складнішими, а засоби захисту повинні відповідати найновішим вимогам кібербезпеки [1].

Впровадження інтелектуальних систем моніторингу мережевого трафіку, які можуть оперативно виявляти аномалії, що свідчать про зовнішнє втручання, є необхідною умовою для забезпечення ефективного захисту комп'ютерних систем. Одним із важливих аспектів є застосування алгоритмів машинного навчання та методів аналізу трафіку для автоматичного виявлення загроз. Такі системи можуть не лише виявляти аномалії, а й автоматично реагувати на них, блокуючи підозрілі дії та запобігаючи поширенню шкідливих впливів.

Таким чином, актуальність теми дослідження обумовлена необхідністю розробки сучасних методів і програмних засобів для моніторингу та аналізу

мережевого трафіку з метою виявлення та блокування зовнішніх вторгнень у комп'ютерні системи. Ця проблема є надзвичайно важливою для забезпечення безпеки інформаційних ресурсів у корпоративних мережах, державних установах та у приватному секторі.

Мета роботи: розробити програмний застосунок моніторингу мережевого трафіку з метою виявлення та блокування зовнішнього вторгнення в комп'ютерні системи.

Для досягнення поставленої мети необхідно виконати наступні задачі:

1. Провести аналіз існуючих методів моніторингу мережевого трафіку та виявлення вторгнень у комп'ютерні системи, визначити їх переваги та недоліки;

2. Дослідити сучасні підходи до виявлення аномалій у мережевому трафіку з використанням алгоритмів машинного навчання та визначити найбільш доцільні з них для вирішення завдання виявлення загроз.

3. Розробити програмний застосунок моніторингу мережевого трафіку за допомогою базових функцій *Python* з перехоплення мережевих пакетів і *CNN* для визначення аномалій у вмісті мережевих пакетів, що дасть можливість виявляти та блокувати зовнішні вторгнення в комп'ютерні системи;

4. Провести тестування розробленого програмного застосунку моніторингу мережевого трафіку для виявлення та блокування зовнішнього вторгнення в комп'ютерні системи, що дасть змогу дослідити доцільність використання розробленого застосунку для вирішення поставленої задачі.

Наукова складова цієї кваліфікаційної роботи полягає у розробці нових підходів до аналізу мережевого трафіку для виявлення і блокування зовнішніх загроз. Зокрема, пропонується застосування алгоритмів машинного навчання для автоматичного виявлення аномалій, що дозволяє підвищити точність виявлення атак та мінімізувати кількість хибнопозитивних спрацювань. Також робота передбачає впровадження комбінованого використання сигнатурних та поведінкових методів для виявлення вторгнень. Це дозволить забезпечити виявлення нових та невідомих загроз, які не можуть бути виявлені лише на основі сигнатур, без використання додаткових апаратних ресурсів.

Об'єкт дослідження: процес моніторингу та аналізу мережевого трафіку в комп'ютерних системах з метою забезпечення їх захисту від зовнішніх загроз.

Предмет дослідження: методи моніторингу мережевого трафіку для виявлення і блокування вторгнень у комп'ютерні системи.

Методи дослідження: аналіз сучасних методів моніторингу мережевого трафіку та існуючих засобів захисту комп’ютерних систем для визначення поширених методів виявлення зовнішнього вторгнення в комп'ютерні системи, моделювання роботи системи для визначення ефективності прийнятих рішень для виявлення аномалій, експериментальні дослідження розробленого застосунку на реальних даних для оцінки його доцільності в умовах виявлення різних типів атак.

Наукова новизна роботи полягає у розробці програмного застосунку, який за рахунок аналізу вмісту пакетів забезпечує моніторинг мережевого трафіку в реальному часі та автоматично блокує зовнішні атаки. Такий підхід дозволяє не лише виявляти атаки, але й негайно реагувати на них, що є надзвичайно важливим для захисту критичних систем.

Кваліфікаційна робота спрямована на розв'язання актуальної проблеми кібербезпеки – захисту комп'ютерних систем від зовнішніх загроз через моніторинг мережевого трафіку. Запропоновані методи і програмні засоби дозволять підвищити ефективність виявлення та блокування загроз, що є важливим етапом у забезпеченні безпеки сучасних інформаційних систем.

Практична цінність роботи полягає у створенні програмного застосунку для моніторингу мережевого трафіку, який завдяки інтеграції штучного інтелекту дозволяє виявляти та блокувати загрози без використання додаткових апаратних потужностей.

Результати кваліфікаційної роботи представлено на 3 науково-практичних конференціях:

1) Борщ А.А. Методи моніторингу мережевого трафіку з метою виявлення та блокування зовнішнього вторгнення в комп'ютерні системи. Матеріали міжн. наук. інтернет-конф. «Інформаційне суспільство: технологічні, економічні та технічні аспекти становлення (випуск 92)» (м. Тернопіль, Україна, м. Ополе,

Польща, 12-13 листопада 2024 р.). ГО “Наукова спільнота”, *WSZIA w Opolu*. Тернопіль. 2024. *http://www.konferenciaonline.org.ua/ua/article/id*-1933/

2) Борщ А.А. Проблеми моніторингу мережевих загроз та запобігання вторгненням в комп’ютерні системи. Тези доповідей міжн. наук.-техн. конф. “Інтелектуальні технології лінгвістичного аналізу” (23-24 жовтня 2024 р.). К.: НАУ, 2024. С. 48.

3) Борщ А.А. Метод моніторингу мережевого трафіку для виявлення зовнішнього вторгнення в комп'ютерну систему. Тези доповідей наук.-практ. конф. “Сучасні тенденції розвитку системного програмування” (21-22 листопада 2023 р.). К.: НАУ, 2024. С. 38-39.

Комп'ютерні системи відіграють важливу роль в сучасному інформаційному суспільстві, забезпечуючи обмін даними між пристроями, користувачами та системами через різноманітні мережеві інфраструктури. Вони є основою корпоративних, урядових та приватних мереж, але через свою складність і масштабність стають об'єктом для атак. Основним джерелом багатьох загроз є мережевий трафік, через який здійснюються спроби вторгнення, викрадення даних або нанесення шкоди. Розуміння вразливостей комп’ютерних систем та побудова засобів їх моніторингу є ключовими елементами сучасної кібербезпеки.

1.1. Принципи виявлення вторгнень в комп’ютерну систему через мережу

Принципи виявлення вторгнень в комп'ютерні системи та мережі ґрунтуються на аналізі мережевого трафіку, поведінки користувачів, даних і системних подій для виявлення підозрілих або шкідливих дій [2]. Системи виявлення вторгнень (*IDS – Intrusion Detection Systems*) працюють з метою виявлення атак, які можуть бути спрямовані на отримання несанкціонованого доступу, зловживання ресурсами, викрадення інформації або руйнування систем. Основні принципи виявлення вторгнень можна поділити на кілька ключових підходів і методів, які доповнюють одне одного.

1. Сигнатурний аналіз (*Signature-based Detection*). Принцип полягає в порівнянні поточної активності мережі або системи з набором відомих шаблонів або сигнатур атак, які були раніше зафіксовані. Якщо вхідний або вихідний трафік або подія відповідають сигнатурі відомої атаки, система спрацьовує і сигналізує про можливе вторгнення.

Переваги:

– висока швидкість виявлення відомих атак;

– ефективний захист від загроз, що вже мають відомі сигнатури.

Недоліки:

– нездатність виявляти нові або модифіковані атаки, що не мають зареєстрованої сигнатури;

– потреба в регулярному оновленні бази сигнатур для підтримки актуальності.

2. Поведінковий аналіз (*Anomaly-based Detection*) базується на вивченні типової поведінки системи або мережі для визначення відхилень, що можуть свідчити про загрозу (рис. 1.1).

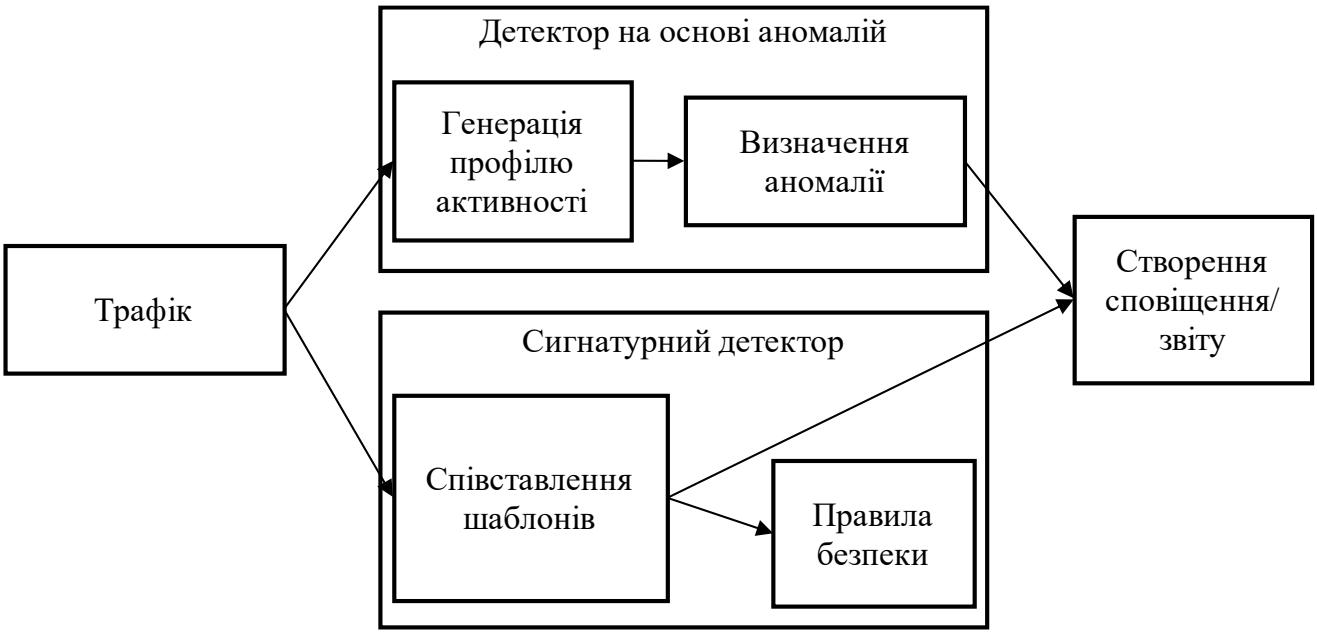

Рис. 1.1. Схема використання сигнатурного та поведінкового аналізу

Система будує модель нормальної поведінки на основі аналізу трафіку, дій користувачів або додатків і фіксує всі відхилення від цієї моделі як потенційні загрози [3].

Переваги:

– здатність виявляти нові та невідомі загрози, що не можуть бути виявлені сигнатурними методами;

– ефективне виявлення аномалій, які можуть свідчити про атаки типу "нульового дня" (*Zero-Day Attacks*).

Недоліки:

– велика кількість хибнопозитивних спрацювань, оскільки будь-яке відхилення від норми може бути помилково інтерпретоване як загроза;

– потреба у постійному оновленні моделей поведінки.

Вбудувати в наявну систему можна наступним чином (рис. 1.2).

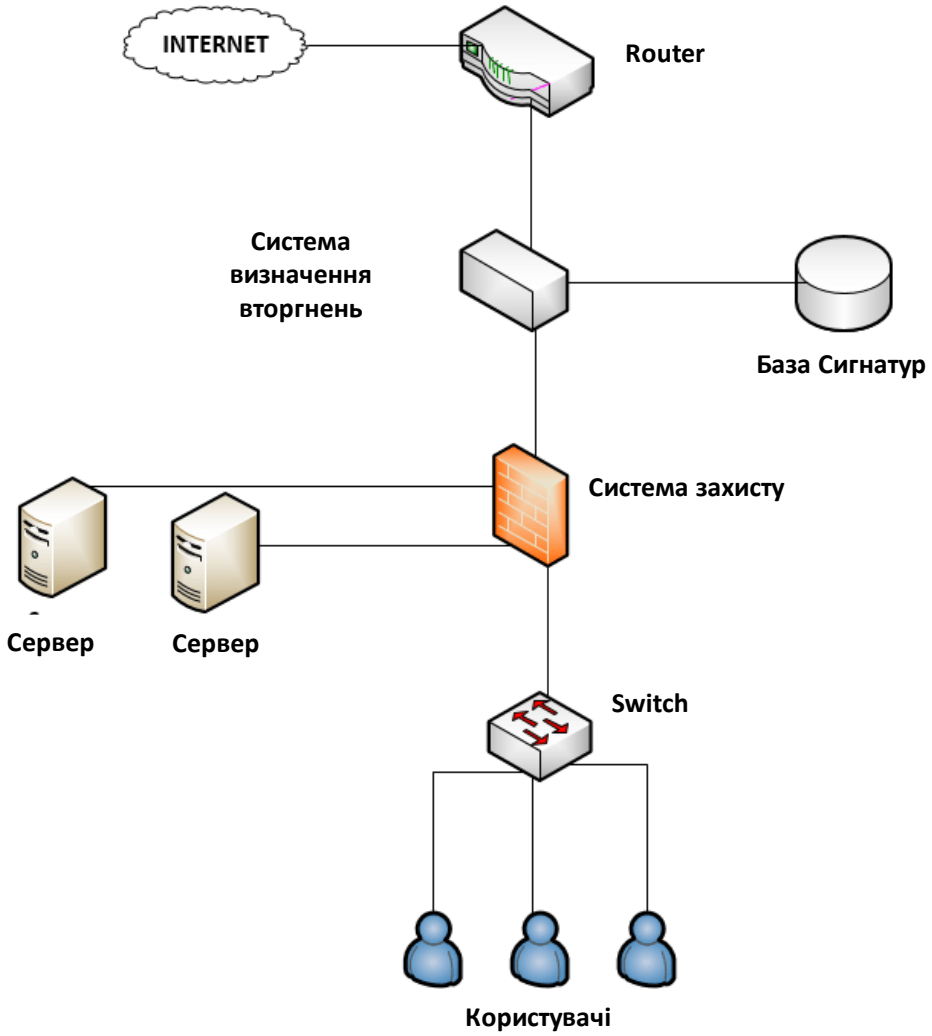

Рис. 1.2. Приклад вбудовування систем додаткового аналізу трафіку в наявні комп’ютерні системи

3. Аналіз цілісності (*Integrity Checking*) – використовується для виявлення змін у критичних файлах або системних конфігураціях. Системи виявлення вторгнень регулярно перевіряють контрольні суми файлів або параметри конфігурації та порівнюють їх з еталонними значеннями. Якщо виявляються невідповідності, це може свідчити про спробу злому або втручання в систему.

Переваги:

– ефективний метод виявлення несанкціонованих змін у критичних системах або файлах.

Недоліки:

– не завжди можна виявити вторгнення в реальному часі;

– може бути складним у реалізації в системах із великим обсягом файлів та постійними змінами.

4. Аналіз логів (*Log-based Detection*) – передбачає збір, аналіз і кореляцію системних журналів (логів) для виявлення підозрілих дій або подій. Логи можуть містити інформацію про входи користувачів, дії програм або трафік між пристроями, що дозволяє відстежувати підозрілу активність і виявляти потенційні загрози [4].

Переваги:

– може виявляти складні атаки через аналіз подій на різних рівнях системи;

– ефективний для розслідування інцидентів після їхнього виникнення.

Недоліки:

– системи виявлення на основі логів можуть працювати з затримкою;

– велика кількість даних для обробки може ускладнити своєчасний аналіз і реагування.

5. Метод кореляції подій (*Event Correlation Analysis*)

Метод кореляції подій передбачає збір інформації з різних джерел і систем для створення єдиної картини подій у мережі.

Цей підхід дозволяє системам *IDS* не лише виявляти окремі інциденти, але й зв'язувати їх між собою, щоб краще розуміти загальну загрозу. Наприклад, атака може починатися з мережевого сканування і завершуватися спробою проникнення у внутрішню систему. Система кореляції може виявити всі етапи атаки [5].

Переваги:

– забезпечує глибокий контекст для виявлення складних атак;

– дає змогу об'єднати події в різних сегментах мережі для кращого розуміння загрози.

Недоліки:

– потребує складної інфраструктури для збору даних і потужних обчислювальних ресурсів для їх аналізу;

– може бути повільнішим порівняно з методами виявлення в реальному часі.

6. Аналіз потоків трафіку (*Flow-based Detection*)

Цей метод передбачає аналіз статистичних даних про потоки мережевого трафіку замість детального аналізу його вмісту. Інструменти на основі аналізу потоків (наприклад, *NetFlow*) збирають дані про кількість переданих пакетів, *IP*-адреси джерела та призначення, порти та інші параметри, що дозволяє виявляти аномалії, характерні для *DDoS*-атак або сканування портів.

Переваги:

– можливість моніторингу великого обсягу трафіку без глибокого аналізу окремих пакетів;

– виявлення аномалій у загальному трафіку, які можуть свідчити про підготовку або реалізацію атак.

Недоліки:

– не забезпечує глибокого аналізу вмісту трафіку;

– може бути неефективним для виявлення складних загроз на рівні додатків.

7. Аналіз на основі машинного навчання (*Machine Learning-based Detection*)

Застосування методів машинного навчання для виявлення вторгнень стає дедалі популярнішим, оскільки ці методи дозволяють аналізувати великі обсяги даних і виявляти складні патерни, які важко помітити за допомогою традиційних методів [6]. Машинне навчання може використовуватися для поведінкового аналізу, класифікації загроз та прогнозування потенційних атак на основі історичних даних.

Переваги:

– висока адаптивність до нових типів загроз;

– можливість виявляти складні атаки, аналізуючи великі обсяги трафіку та подій.

Недоліки:

– потреба в навчанні системи на великих обсягах даних, що може вимагати значних ресурсів;

– ризик хибних позитивних або негативних спрацювань залежно від якості навчання.

Моніторинг і аналіз мережевого трафіку є важливим елементом кібербезпеки в корпоративних мережах, оскільки дозволяє своєчасно виявляти аномалії та можливі загрози. Існує декілька основних принципів, яких дотримуються при моніторингу та аналізі мережі для виявлення загроз.

1. Комплексний підхід до моніторингу

Ефективний моніторинг мережевого трафіку повинен охоплювати всі рівні корпоративної мережі, включаючи мережеву інфраструктуру (маршрутизатори, комутатори, брандмауери), сервери, робочі станції та мобільні пристрої. Такий підхід дозволяє отримати повну картину активності у мережі та швидко виявляти можливі загрози. Комплексний моніторинг передбачає збір та аналіз даних з усіх вузлів мережі, що включає не тільки вхідний та вихідний трафік, але й внутрішні зв'язки між різними компонентами мережі [7].

2. Аналіз аномалій

Одним з ключових методів виявлення загроз є аналіз аномалій у мережевому трафіку. Цей метод полягає в побудові моделі нормальної поведінки мережі та виявленні відхилень від цієї моделі. Аномальні дії можуть свідчити про спробу вторгнення або інші шкідливі дії. Наприклад, раптове збільшення кількості запитів до сервера може свідчити про *DDoS*-атаку.

Для побудови моделей нормальної поведінки використовуються різні методи, зокрема машинне навчання, яке дозволяє системам самостійно навчатися і адаптуватися до змін у мережевому трафіку. Це дозволяє виявляти нові типи атак, які не можуть бути розпізнані за допомогою традиційних сигнатурних методів [8].

3. Сигнатурний аналіз

Сигнатурний аналіз є класичним методом виявлення загроз, який полягає у порівнянні мережевого трафіку з базою даних відомих загроз. Кожна загроза має

унікальний шаблон (сигнатуру), за яким її можна виявити. Системи виявлення загроз (*Intrusion Detection Systems, IDS*) та системи запобігання вторгненням (*Intrusion Prevention Systems, IPS*) використовують сигнатурний аналіз для швидкого виявлення відомих атак.

Проте цей метод має обмеження, оскільки не дозволяє виявляти нові або модифіковані атаки, які ще не мають відповідних сигнатур у базі даних. Для подолання цього обмеження сигнатурний аналіз часто поєднується з поведінковим аналізом або аналізом аномалій.

4. Поведінковий аналіз

Поведінковий аналіз базується на вивченні дій користувачів або пристроїв у мережі та виявленні відхилень від їх звичайної поведінки. Наприклад, якщо користувач зазвичай працює в певний час і доступ до системи з певних локацій, спроба входу до системи з іншого місця у незвичний час може бути сигналом про компрометацію облікового запису [9].

Поведінковий аналіз є важливим інструментом для виявлення атак на рівні користувачів та внутрішніх загроз. Він дозволяє визначити нестандартні дії, що можуть свідчити про підготовку або проведення атаки.

5. Реальний час і реакція на загрози

Моніторинг мережевого трафіку в режимі реального часу є ключовою умовою для оперативного виявлення та реагування на загрози. Системи кібербезпеки повинні не лише виявляти атаки, але й негайно блокувати або ізолювати підозрілі дії. Це дозволяє мінімізувати наслідки атак та зменшити ризики для критичних систем [10].

Сучасні системи моніторингу та захисту мережі використовують автоматизовані механізми для реагування на загрози, що включають автоматичне блокування *IP*-адрес, ізоляцію скомпрометованих вузлів та інші заходи безпеки.

6. Візуалізація даних і звітність

Для ефективного моніторингу мережі важливо не лише збирати і аналізувати дані, але й забезпечувати їхню зручну візуалізацію. Графіки, діаграми та інтерактивні інструменти дозволяють адміністраторам мережі швидко

оцінювати стан безпеки та виявляти потенційні загрози. Візуалізація також дозволяє зосередитися на аномаліях і здійснювати швидке розслідування інцидентів без необхідності переглядати великі масиви даних.

7. Безперервний моніторинг

Оскільки атаки можуть відбуватися у будь-який момент часу, важливим принципом є безперервний моніторинг мережевого трафіку. Це забезпечує постійне спостереження за станом мережі та оперативне виявлення будь-яких змін у трафіку, які можуть свідчити про спроби вторгнення.

Безперервний моніторинг також дозволяє вчасно помічати тенденції, що можуть свідчити про підготовку до атаки, та вживати превентивних заходів.

Вразливості комп’ютерних систем, що виникають через мережевий трафік, є серйозною загрозою для сучасних організацій та їхніх інфраструктур. Важливими аспектами забезпечення безпеки є моніторинг і аналіз трафіку, який дозволяє виявляти та блокувати загрози на ранніх етапах.

1.2. Аналіз загроз і необхідність контролю мережевого трафіку для захисту комп’ютерних систем

З розвитком інформаційних технологій і зростанням кількості підключених до мережі пристроїв, проблеми кібербезпеки стають дедалі актуальнішими. Зокрема, одним із ключових завдань кібербезпеки є захист комп'ютерних систем від різноманітних мережевих атак, які можуть призвести до збоїв у роботі, втрати даних або навіть компрометації всієї мережевої інфраструктури. Контроль мережевого трафіку є важливим заходом для виявлення та нейтралізації загроз, що виникають у процесі функціонування комп'ютерних систем.

З розвитком *IoT* і хмарних технологій зростає кількість підключених пристроїв, які генерують величезний обсяг даних. Це збільшує навантаження на мережеву інфраструктуру та ускладнює моніторинг мережевого трафіку. Традиційні методи аналізу не завжди дозволяють виявити складні атаки, особливо ті, що маскуються під легітимний трафік. У зв’язку з цим виникає потреба у

впровадженні більш інтелектуальних підходів до аналізу трафіку, таких як використання машинного навчання та поведінкових моделей.

Одним із ключових аспектів актуальності є зростання частоти цільових атак на комп’ютерні системи державних установ, фінансових організацій та приватного бізнесу. Це призводить до значних фінансових втрат, витоку конфіденційної інформації та порушення безперервності бізнес-процесів. Ефективний контроль мережевого трафіку може не лише запобігти атакам, але й забезпечити своєчасне реагування на інциденти, що вже відбулися.

Також важливим є зростання популярності розподілених систем і мікросервісної архітектури, де велика кількість вузлів обмінюється даними в режимі реального часу. У таких умовах кожен елемент системи стає потенційною точкою входу для зловмисників. Контроль трафіку у таких системах дозволяє забезпечити цілісність комунікації та уникнути поширення атак через взаємопов’язані елементи.

Швидкий розвиток регуляторних вимог, таких як *GDPR* та *ISO* 27001, накладає додаткові зобов’язання на організації у частині захисту даних і забезпечення прозорості роботи їхніх систем. Відсутність належного контролю мережевого трафіку може призвести до юридичних санкцій і втрати довіри клієнтів. Це додає ще одного рівня актуальності дослідженню методів моніторингу та захисту мереж.

Аналіз загроз і контроль мережевого трафіку є не лише необхідними заходами для забезпечення безпеки комп’ютерних систем, але й фундаментом для стійкого функціонування цифрової інфраструктури в умовах сучасних викликів.

1.2.1. Типи мережевих атак: *DDoS, DoS, MITM* та інші

Мережеві атаки – це спроби отримати несанкціонований доступ до комп'ютерних систем або порушити їхню роботу шляхом маніпуляції мережевим трафіком. Існує багато різновидів атак, які можуть здійснюватися зловмисниками

з різною метою, і їх умовно можна поділити на кілька категорій. Ось основні типи мережевих атак:

1. Атаки типу відмовлення в обслуговуванні (*DoS*)

DoS (*Denial of Service*) – це атаки, спрямовані на виведення з ладу системи або мережевого ресурсу, заповнюючи його надмірною кількістю запитів або використовуючи всі доступні ресурси (наприклад, процесорні потужності або смугу пропускання). В результаті, легітимні користувачі не можуть отримати доступ до ресурсів, оскільки система стає перевантаженою.

Основна мета *DoS*-атак полягає у порушенні нормального функціонування сервісу або сервера. Атаки можуть бути спрямовані як на фізичні сервери, так і на мережеві вузли, пристрої або програмні ресурси. Використання таких атак може призвести до втрати доходу для компанії, порушення її репутації або просто створення незручностей для кінцевих користувачів.

До основних різновидів *DoS*-атак можна віднести:

– *UDP*-флуд (*UDP flood*): атака за допомогою надсилання великої кількості датаграм *UDP* (*User Datagram Protocol*) на певний порт системи, що призводить до споживання ресурсів і викликає перевантаження системи;

– *SYN*-флуд (*SYN flood*): атака спрямована на використання протоколу *TCP*, де зловмисник надсилає багато запитів на встановлення з’єднання, але не завершує тристоронній обмін. У результаті сервер витрачає ресурси на зберігання відкритих з'єднань, які ніколи не будуть завершені.

2. Розподілені атаки відмовлення в обслуговуванні (*DDoS*)

DDoS (*Distributed Denial of Service*) – це різновид *DoS*-атак, але з тією відмінністю, що атака здійснюється не з одного джерела, а з кількох одночасно, що робить її набагато складнішою для блокування (рис. 1.3). Зловмисник керує великою кількістю заражених комп'ютерів або пристроїв (ботнетів), які одночасно надсилають велику кількість запитів до жертви, перевантажуючи її ресурси [11].

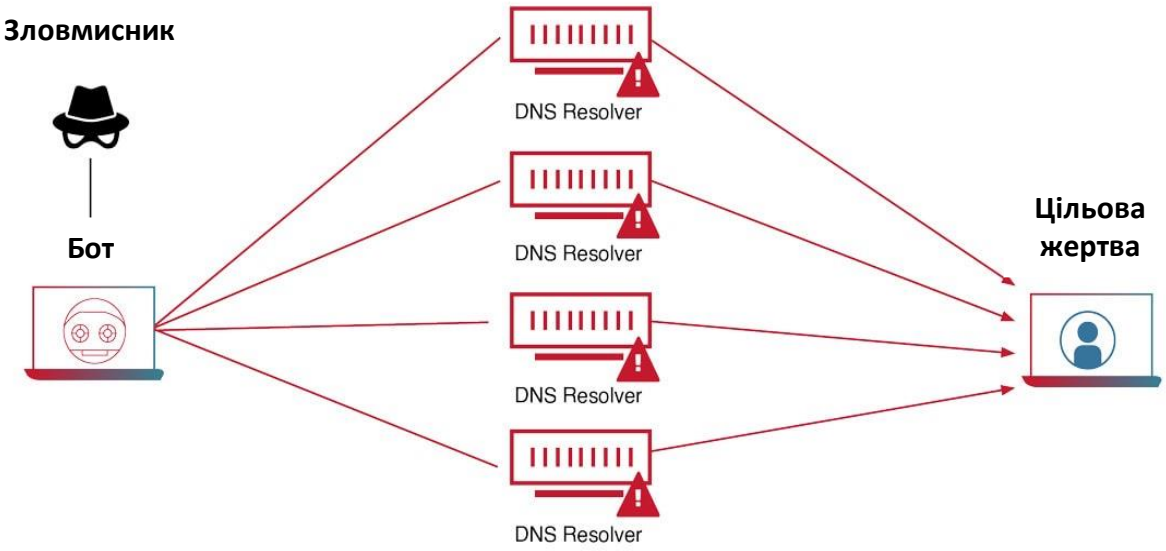

Рис. 1.3. Принцип дії атаки типу *DDoS*

DDoS-атаки є значно небезпечнішими, ніж *DoS*, оскільки їхні наслідки можуть бути масштабнішими і складнішими для виявлення. Використання багатьох джерел у різних місцях ускладнює блокування атак на рівні фільтрації *IP*-адрес або мережевого трафіку.

3. Атаки типу "людина посередині" (*MITM*)

MITM (*Man-in-the-Middle*) – це атака, за якої зловмисник стає посередником між двома легітимними користувачами або пристроями в мережі. Це дозволяє йому перехоплювати, змінювати або записувати інформацію, яка передається між ними, без їх відома. Атаки *MITM* можуть бути здійснені у різних формах, включаючи:

– *ARP*-спуфінг (*ARP spoofing*): зловмисник надсилає підроблені *ARP*-повідомлення в мережі, щоб перехопити трафік, який має йти до іншого пристрою;

– *DNS*-спуфінг (*DNS spoofing*): атака, за якої зловмисник підробляє записи *DNS* для того, щоб перенаправляти трафік на підконтрольний йому сервер (рис. 1.4).

Рис. 1.4. Принцип дії *MITM*-атаки

MITM-атаки є особливо небезпечними в корпоративних мережах, де зловмисник може отримати доступ до конфіденційної інформації або модифікувати дані без виявлення. Вони можуть використовуватися для крадіжки паролів, номерів кредитних карток або інших чутливих даних.

4. Атаки типу "фішинг" (*Phishing*)

Фішинг – це вид соціальної інженерії, за якого зловмисник намагається обманом змусити користувача розкрити конфіденційну інформацію, таку як паролі або фінансові дані. Це може бути зроблено через підроблені електронні листи, вебсайти або навіть телефонні дзвінки. Метою фішингових атак є викрадення особистої інформації або встановлення шкідливого програмного забезпечення на комп'ютері користувача.

Фішингові атаки часто супроводжуються іншими видами атак, наприклад, *MITM* або зараженням шкідливими програмами. Крім того, вони використовують психологічні методи впливу, такі як нагнітання страху або терміновості, щоб змусити жертву виконати необхідні дії.

5. Шкідливе програмне забезпечення (*Malware*)

Malware – це загальна назва для різноманітних шкідливих програм, які можуть бути використані для отримання доступу до системи, пошкодження даних або порушення нормальної роботи комп'ютерів. До основних типів шкідливого ПЗ належать:

– Трояни (*Trojans*): програми, що маскуються під легітимні додатки, але насправді виконують шкідливі дії;

– Віруси (*Viruses*): шкідливі програми, які поширюються шляхом зараження інших файлів або програм;

– Хробаки (*Worms*): програми, які самостійно розмножуються у мережі, не потребуючи взаємодії з користувачем.

Шкідливе ПЗ може бути встановлене через мережу, зазвичай за допомогою фішингових атак або інших соціальних інженерних методів.

6. Атаки на веб-сервіси (*SQL*-ін'єкції, *XSS*)

SQL-ін'єкції (*SQL Injection*) і міжсайтове сценаріювання (*Cross-Site Scripting, XSS*) є одними з найпоширеніших атак на веб-додатки. Вони використовують уразливості в програмному коді веб-додатків для введення шкідливих команд у бази даних або для виконання шкідливого сценарію на стороні клієнта;

– *SQL*-ін'єкція: полягає у введенні шкідливих *SQL*-запитів у поля для введення даних на веб-сайті, що дозволяє отримати доступ до баз даних або змінювати дані в них;

– *XSS*: полягає у введенні шкідливого сценарію (*JavaScript*) у веб-сторінки, який виконується на пристрої користувача, що відвідує цю сторінку.

1.2.2. Приклади *DDoS*-атак на критичні системи

Атаки типу *DDoS* (*Distributed Denial of Service*) є одними з найпоширеніших і найнебезпечніших загроз для мережевих інфраструктур і критичних систем у всьому світі. Метою *DDoS*-атак є перевантаження цільової системи великою кількістю запитів або спроб з’єднання з різних джерел, що призводить до неможливості виконання легітимних запитів користувачів. Ці атаки часто використовують ботнети – мережі заражених комп'ютерів або пристроїв, керовані зловмисниками для координації атаки. *DDoS*-атаки можуть мати серйозні наслідки для критичних систем, що забезпечують функціонування банківських, державних, медичних і промислових установ [8]:

1. Атака на *GitHub* (2018 рік). Один із найбільш показових випадків *DDoS*-атак стався у лютому 2018 року, коли *GitHub* – один з найбільших сервісів для спільної розробки програмного забезпечення – зазнав наймасштабнішої *DDoS*-атаки в історії на той час. Атака досягла пікової потужності 1,35 Тбіт/с, що перевищило більшість попередніх відомих атак.

У цьому випадку було використано метод ампліфікації *Memcached*, коли атакуючі використовували незахищені сервери *Memcached* для підсилення своїх запитів. *Memcached* – це система кешування, яка зазвичай використовується для прискорення веб-додатків шляхом зменшення навантаження на бази даних. Зловмисники відправляли малі запити на сервери, які відповідали надзвичайно великими обсягами даних на адресу *GitHub*, що призвело до значного навантаження на інфраструктуру сайту.

GitHub швидко реагував, перенаправивши трафік через сервіс *DDoS*-захисту *Akamai*, який допоміг поглинути трафік і відновити нормальну роботу протягом декількох хвилин. Цей випадок продемонстрував як масштаби сучасних *DDoS*-атак, так і важливість готовності до них з боку критично важливих інтернет-сервісів.

2. Атака на *Dyn* (2016 рік). У жовтні 2016 року відбулася масштабна *DDoS*-атака на компанію *Dyn*, яка надає послуги *DNS* для багатьох великих веб-сервісів, зокрема *Twitter*, *Netflix*, *Reddit*, *PayPal* і *Spotify*. Атака використовувала ботнет, побудований на основі шкідливого програмного забезпечення *Mirai*, яке заражало пристрої Інтернету речей (*IoT*) – зокрема, камери спостереження, маршрутизатори і цифрові відеореєстратори.

Атака мала кілька хвиль і досягла пікової потужності понад 1 Тбіт/с. Використовуючи заражені пристрої *IoT*, зловмисники відправляли величезну кількість запитів до серверів *Dyn*, перевантажуючи їх і роблячи *DNS*-сервіси недоступними для багатьох популярних інтернет-сервісів. Це призвело до того, що значна частина користувачів у Північній Америці та Європі не могла отримати доступ до популярних веб-сайтів протягом декількох годин.

Цей інцидент привернув увагу до загрози, яку представляють незахищені *IoT*-пристрої, оскільки *Mirai* використовував їх у величезній кількості для проведення атаки. Атака на *Dyn* також підкреслила важливість забезпечення надійного захисту *DNS*-систем, оскільки *DNS* є критично важливою частиною інтернет-інфраструктури.

3. Атака на *Estonian Government Infrastructure* (2007 рік). У 2007 році Естонія зазнала серії масштабних *DDoS*-атак, які стали одним із перших великих прикладів кібервійни проти державної інфраструктури. Після політичних заворушень, пов’язаних з переміщенням радянського пам'ятника, урядові сайти, медіа-платформи, фінансові установи та онлайн-сервіси Естонії були атаковані зловмисниками з різних частин світу.

Атака включала в себе *HTTP*-флуди, *SYN*-флуди та інші методи перевантаження серверів великими обсягами трафіку, що призвело до відмови в обслуговуванні багатьох урядових і комерційних веб-сайтів. Багато критичних державних сервісів були недоступні протягом кількох днів, а фінансові транзакції були значно ускладнені.

Цей інцидент став важливою подією в історії кібербезпеки, оскільки продемонстрував, як кібератаки можуть бути використані як інструмент геополітичного тиску. Він також підштовхнув Естонію до створення однієї з найбільш передових систем кібербезпеки у світі та став основою для формування міжнародних ініціатив у сфері кіберзахисту.

4. Атака на *Amazon Web Services* (*AWS*) (2020 рік). У лютому 2020 року *Amazon Web Services* (*AWS*), один із найбільших хмарних сервісів у світі, зазнав *DDoS*-атаки з піковою потужністю 2,3 Тбіт/с. Це стало однією з найбільших *DDoS*-атак в історії, і вона використовувала метод ампліфікації *CLDAP* (*Connection-less Lightweight Directory Access Protocol*).

CLDAP – це мережевий протокол, який використовується для роботи з каталогами в мережах, таких як *Active Directory*. Зловмисники використовували незахищені *CLDAP*-сервери для ампліфікації своїх запитів, що дозволило їм створити значний обсяг трафіку, спрямованого на *AWS*. Атака тривала кілька днів,

і хоча *AWS* змогла ефективно протидіяти їй, вона підкреслила зростаючу загрозу надпотужних *DDoS*-атак.

Цей випадок демонструє, як атакуючі можуть використовувати уразливі інфраструктурні сервіси для проведення надзвичайно потужних атак. Він також підкреслює важливість забезпечення надійного захисту хмарних сервісів, які все частіше стають мішенню для зловмисників.

5. Атака на Банки Південної Кореї та США (2011-2013 роки). У період між 2011 і 2013 роками кілька великих банків Південної Кореї та США зазнали серії *DDoS*-атак, які мали на меті порушити їхню роботу та дестабілізувати фінансові системи цих країн. Атаки були спрямовані на банківські веб-сайти та внутрішні інфраструктури, що призвело до значних збоїв у роботі онлайн-банкінгу та обробці фінансових транзакцій.

Однією з найбільш відомих атак цього періоду стала атака на *South Korean Nonghyup Bank* у 2011 році, яка тривала кілька годин і порушила роботу банкоматів, онлайн-банкінгу та внутрішніх серверів. Це призвело до того, що мільйони клієнтів не могли отримати доступ до своїх рахунків або здійснювати фінансові операції.

У 2013 році аналогічні атаки були спрямовані на великі американські банки, зокрема *Bank of America*, *JP Morgan Chase* та *Wells Fargo*, використовуючи техніку ампліфікації *DNS*-запитів. Хоча ці атаки не призвели до тривалих збоїв, вони спричинили значні тимчасові порушення в роботі систем і підвищили обізнаність про важливість захисту фінансових установ від *DDoS*-загроз.

Приклади *DDoS*-атак на критичні системи свідчать про те, що такі атаки можуть мати серйозні наслідки для державних, комерційних і фінансових інфраструктур. Від атак на хмарні сервіси та *DNS*-системи до атак на державні установи та банки, *DDoS*-атаки є потужним інструментом, який може спричинити значні фінансові збитки та порушення роботи систем. Захист від таких атак є критично важливим для забезпечення безперебійної роботи важливих інфраструктур і зменшення впливу на користувачів та економіку.

1.2.3. Аналіз ефективності заходів протидії мережевим атакам

У сучасному світі мережеві атаки стають дедалі поширенішими, різноманітнішими та небезпечнішими. З розвитком Інтернету та мережевих технологій збільшується й кількість уразливостей, які можуть бути використані зловмисниками для проведення атак на комп'ютерні системи. Для забезпечення безпеки інформаційних систем і мінімізації загроз важливо не лише вчасно виявляти атаки, а й ефективно їм протидіяти. Протидія мережевим атакам включає використання різних технологій, методів та інструментів, які допомагають захистити системи від вторгнень і шкідливого трафіку. У цьому розділі ми розглянемо основні заходи протидії атакам та їх ефективність.

1. Системи виявлення та запобігання вторгненням (*IDS*/*IPS*)

Одним із найбільш поширених інструментів для протидії мережевим атакам є системи виявлення та запобігання вторгненням (*IDS*/*IPS*). *Intrusion Detection Systems* (*IDS*) призначені для моніторингу мережевого трафіку або дій користувачів у системах з метою виявлення підозрілої активності, що може свідчити про вторгнення або кібератаку. *Intrusion Prevention Systems* (*IPS*) є подальшим розвитком *IDS* і не лише виявляють атаки, але й вживають заходів для їх запобігання, таких як блокування трафіку або ізоляція сегменту мережі.

Ефективність *IDS*/*IPS* полягає в здатності швидко ідентифікувати загрози та запобігати їм у реальному часі. Багато сучасних *IDS*/*IPS* систем використовують сигнатурний аналіз для виявлення відомих типів атак, що дозволяє вчасно блокувати відомі загрози. Проте, однією з проблем є те, що сигнатурні системи не здатні виявляти нові або модифіковані атаки, для яких ще немає відповідних сигнатур.

Останнім часом багато *IDS*/*IPS* систем почали використовувати поведінковий аналіз та машинне навчання для виявлення аномалій у трафіку, що дозволяє їм ефективніше боротися з новими та складними загрозами. Таким чином, використання поведінкових методів у поєднанні з традиційним сигнатурним підходом дозволяє підвищити ефективність систем захисту.

2. Фаєрволи (брандмауери)

Фаєрволи (*firewalls*) є ще одним важливим інструментом захисту мереж. Вони працюють на основі набору правил, які визначають, який трафік може проходити через мережу, а який має бути заблокований. Фаєрволи можуть працювати на різних рівнях, зокрема на рівні *IP*-адрес, портів, додатків тощо.

Ефективність фаєрволів залежить від правильної конфігурації та актуальності правил безпеки. Якщо правила налаштовані неправильно, це може призвести до пропускання небезпечного трафіку або блокування легітимного. Сучасні фаєрволи, такі як *NGFW* (*Next-Generation Firewalls*), включають додаткові функції, зокрема виявлення атак, контроль додатків, шифрування трафіку та інтеграцію з *IDS*/*IPS*, що робить їх більш ефективними для протидії складним атакам.

Однак, фаєрволи не завжди можуть захистити від внутрішніх загроз або складних багатостадійних атак, які можуть використовувати різні шляхи для проникнення у мережу. Таким чином, фаєрволи є ефективним інструментом, але їх слід використовувати в поєднанні з іншими засобами безпеки.

3. Системи запобігання *DDoS*-атакам

Однією з найсерйозніших загроз для мережевих систем є *DDoS*-атаки (*Distributed Denial of Service*). Для запобігання таким атакам використовуються спеціальні інструменти та сервіси, зокрема системи захисту від *DDoS*, які можуть виявляти і блокувати підозрілий трафік, що надходить від великої кількості джерел.

Системи захисту від *DDoS*-атак працюють за рахунок кількох ключових методів:

– Аналіз трафіку: виявлення аномалій у мережевому трафіку, таких як раптове збільшення кількості запитів або незвичайні шаблони трафіку;

– Фільтрація трафіку: блокування підозрілих запитів або перенаправлення їх через спеціальні сервіси для поглинання атаки;

– Розподіл навантаження: використання географічно розподілених центрів обробки даних або мережі доставки контенту (*CDN*), щоб розподілити навантаження на кілька серверів.

Ефективність захисту від *DDoS* залежить від здатності систем швидко виявляти та ізолювати шкідливий трафік. Однак такі системи мають обмеження, якщо атака надто велика або якщо використовується нова техніка атак, яку система не може виявити. Крім того, ефективність систем залежить від пропускної здатності інфраструктури, що може бути обмеженням у випадку надзвичайно потужних атак.

4. Шифрування та *VPN*

Шифрування трафіку є важливим заходом для захисту мережевих даних від перехоплення та модифікації. Використання протоколів *SSL*/*TLS* для захищеного з'єднання дозволяє забезпечити конфіденційність та цілісність переданих даних між клієнтами та серверами.

Крім того, віртуальні приватні мережі (*VPN*) використовуються для створення захищених тунелів між користувачами та внутрішніми мережами компанії, що захищає трафік від несанкціонованого доступу. *VPN* також є ефективним способом захисту підключень до публічних мереж, таких як *Wi-Fi*, оскільки вони шифрують усі передані дані.

Однак шифрування та *VPN* не можуть повністю захистити від усіх загроз, зокрема від *DDoS*-атак, оскільки вони не блокують обсяг трафіку, що надходить на сервер. Крім того, системи моніторингу можуть мати труднощі з аналізом зашифрованого трафіку, що ускладнює виявлення аномалій.

5. Сегментація мережі

Сегментація мережі – це один з ефективних методів запобігання поширенню атак у внутрішній інфраструктурі компанії. Цей метод передбачає поділ мережі на кілька сегментів, кожен з яких має свої правила доступу і безпеки. Таким чином, якщо одна частина мережі піддається атаці або компрометації, інші сегменти залишаються захищеними.

Ефективність сегментації полягає в обмеженні доступу зловмисників до критичних ресурсів у разі порушення безпеки. Наприклад, доступ до серверів з фінансовими даними можна обмежити лише для певних користувачів і служб. Сегментація також допомагає виявляти аномалії в трафіку між сегментами, що може свідчити про можливу атаку.

Однак сегментація мережі потребує ретельного планування та налаштування, оскільки неправильна конфігурація може створити додаткові уразливості. Крім того, великий рівень сегментації може ускладнити адміністрування мережі.

6. Навчання персоналу та підвищення обізнаності

Одним з найбільш ефективних заходів протидії мережевим атакам є навчання персоналу. Навіть найсучасніші технології захисту можуть бути неефективними, якщо користувачі не знають про базові принципи безпеки або нехтують ними. Зловмисники часто використовують соціальну інженерію, фішинг та інші техніки для того, щоб обманом змусити користувачів надати доступ до мережі або встановити шкідливе ПЗ.

Навчання та підвищення обізнаності про загрози є важливим компонентом кібербезпеки. Регулярні тренінги з безпеки, симуляції фішингових атак та створення політик безпеки допомагають мінімізувати ризик людської помилки, що може призвести до вторгнення в мережу.

7. Оцінка ефективності заходів

Для забезпечення ефективного захисту від мережевих атак важливо не лише впроваджувати засоби захисту, але й постійно оцінювати їхню ефективність. Це можна зробити за допомогою періодичних тестувань безпеки, таких як пентестування (*penetration testing*), коли фахівці з кібербезпеки імітують атаки на систему, щоб виявити слабкі місця.

Крім того, важливо проводити моніторинг і аудит систем безпеки для виявлення нових загроз та недоліків у захисті. Системи збору та аналізу журналів подій (*SIEM*) можуть бути корисними для відстеження та аналізу інцидентів безпеки.

Ефективність заходів протидії мережевим атакам залежить від поєднання різних методів і технологій. Системи *IDS*/*IPS*, фаєрволи, захист від *DDoS*-атак, шифрування трафіку, сегментація мережі та навчання персоналу – все це є важливими елементами комплексної стратегії кібербезпеки. Проте жоден із цих інструментів не може забезпечити повний захист окремо. Тому ефективною є тільки багаторівнева система безпеки, яка включає комбінацію технологічних засобів і людських факторів.

1.3. Проблеми моніторингу мережевих загроз та запобігання вторгненням в комп’ютерні системи

Моніторинг мережевого трафіку та запобігання вторгненням в комп'ютерні системи є одними з найважливіших завдань кібербезпеки. З огляду на постійне збільшення обсягів мережевого трафіку, а також постійно зростаючу складність мережевих загроз, проблема забезпечення ефективного моніторингу та запобігання стає дедалі актуальнішою. Основна мета цих процесів полягає у своєчасному виявленні аномальної активності, запобіганні атакам і мінімізації ризиків, пов'язаних з кібератаками. Проте існує низка проблем, пов'язаних із технологіями моніторингу та захисту, що ускладнюють цей процес.

1. Складність сучасних мереж

Сучасні комп'ютерні мережі, зокрема корпоративні мережі великих підприємств, є надзвичайно складними. Вони включають безліч різноманітних компонентів, серед яких локальні мережі (*LAN*), глобальні мережі (*WAN*), хмарні інфраструктури, мобільні пристрої, *IoT*-пристрої тощо. Кожен з цих компонентів має свої специфічні особливості, зокрема різні протоколи передачі даних, рівні доступу, а також уразливості.

Один з основних викликів, з якими стикаються фахівці з безпеки, полягає в тому, що необхідно забезпечити моніторинг усіх цих компонентів, включаючи як внутрішні, так і зовнішні зв'язки. Це потребує інтеграції різноманітних інструментів для моніторингу, таких як системи виявлення та запобігання

вторгнень (*IDS*/*IPS*), фаєрволи, системи аналізу трафіку, а також засоби для контролю доступу до мережі.

Інша складність полягає в тому, що багато компаній використовують хмарні сервіси, що ускладнює завдання моніторингу. З одного боку, хмарні сервіси надають доступ до потужних інструментів обробки даних і можуть спростити управління інфраструктурою. З іншого боку, використання хмарних рішень створює нові проблеми щодо захисту даних та забезпечення контролю за безпекою, оскільки частина інфраструктури може бути розміщена поза межами контролю підприємства.

2. Проблеми збору та аналізу великих обсягів даних

Одна з найголовніших проблем, пов'язаних з моніторингом мережевого трафіку, полягає в обробці великих обсягів даних. Сучасні корпоративні мережі генерують величезні обсяги трафіку, які повинні бути постійно аналізовані для виявлення потенційних загроз. Збирання всіх цих даних, їх зберігання та аналіз вимагають значних ресурсів, як апаратних, так і програмних.

Наслідком цього є те, що система моніторингу повинна бути не тільки потужною, але й оптимізованою для роботи з великими обсягами даних. Існують такі виклики:

– масштабованість: система моніторингу повинна мати можливість обробляти дедалі більші обсяги даних без зниження продуктивності;

– продуктивність: важливо, щоб система могла аналізувати трафік у реальному часі та своєчасно виявляти загрози;

– фільтрація та агрегація: для зменшення обсягу оброблюваних даних система повинна мати можливість фільтрувати непотрібні дані та агрегувати схожі події.

Без ефективного вирішення цих проблем система моніторингу може стати перевантаженою, що призведе до затримок у виявленні загроз або навіть до неможливості виявлення аномальної активності вчасно.

3. Висока кількість хибнопозитивних спрацювань

Однією з основних проблем, з якою стикаються системи моніторингу загроз, є висока кількість хибнопозитивних спрацювань. Хибнопозитивні спрацювання виникають, коли система виявляє легітимні дії як загрози. Це може відбуватися через невідповідність між конфігурацією системи моніторингу та реальними робочими процесами в мережі.

Висока кількість хибнопозитивних спрацювань може мати кілька негативних наслідків:

– Зниження ефективності: фахівці з кібербезпеки змушені витрачати багато часу на перевірку хибнопозитивних інцидентів замість того, щоб зосередитися на реальних загрозах;

– Ігнорування попереджень: якщо система постійно генерує велику кількість хибних попереджень, існує ризик, що важливі повідомлення можуть бути проігноровані;

– Витрати на адміністрування: високий рівень хибнопозитивних спрацювань вимагає постійного налаштування та оптимізації системи моніторингу, що може призвести до збільшення витрат на кібербезпеку.

Для зниження кількості хибнопозитивних спрацювань важливо забезпечити точне налаштування систем моніторингу, враховуючи специфіку мережі, типи трафіку та типові дії користувачів.

4. Динамічність загроз

Оскільки нові загрози з'являються постійно, система моніторингу повинна бути здатною адаптуватися до змін. Кіберзлочинці використовують різні методи для обходу традиційних заходів захисту, зокрема змінюючи свої стратегії атаки, використовуючи нові експлойти або модифікуючи наявні методи.

Динамічність загроз вимагає від системи моніторингу здатності до навчання та адаптації. Один із способів вирішення цієї проблеми – впровадження методів машинного навчання, які дозволяють системі вивчати поведінку трафіку і самостійно визначати нові типи загроз.

Проте застосування машинного навчання також пов'язане з рядом труднощів:

– Потреба у великих обсягах даних для навчання: системи, засновані на машинному навчанні, вимагають великих обсягів даних для створення моделей. Неправильне навчання може призвести до збільшення хибнопозитивних спрацювань;

– Необхідність постійного оновлення: моделі машинного навчання повинні постійно оновлюватися з урахуванням нових типів загроз.

5. Проблема захисту розподілених мереж і *IoT*

З розвитком Інтернету речей (*IoT*) та збільшенням кількості підключених до мережі пристроїв, захист розподілених мереж стає ще більш складним завданням. *IoT*-пристрої часто мають обмежені можливості з точки зору безпеки, а також не завжди отримують своєчасні оновлення, що робить їх уразливими для атак.

Однією з головних проблем у контексті *IoT* є те, що такі пристрої можуть бути використані як точка входу для атаки на всю мережу. Наприклад, зловмисники можуть отримати доступ до *IoT*-пристрою, а потім використовувати його для здійснення атаки на інші компоненти мережі.

Для вирішення цієї проблеми необхідно впроваджувати спеціалізовані інструменти для моніторингу та захисту *IoT*-пристроїв. Ці інструменти повинні бути здатні виявляти та блокувати підозрілі дії на рівні *IoT*, а також інтегруватися з традиційними системами моніторингу мережевого трафіку.

6. Шифрування трафіку, яке стало стандартною практикою для забезпечення безпеки передавання даних через мережу. Протоколи, такі як *SSL*/*TLS*, забезпечують шифрування трафіку між сервером і клієнтом, що робить його недоступним для перехоплення або модифікації.

Однак шифрування також створює проблеми для систем моніторингу, оскільки воно ускладнює аналіз трафіку. Традиційні системи *IDS*/*IPS* не можуть переглядати вміст зашифрованих повідомлень без доступу до ключів шифрування. Це означає, що багато атак можуть залишитися непоміченими, оскільки шкідливий код може бути захований всередині зашифрованого трафіку.

Щоб вирішити цю проблему, деякі системи моніторингу використовують методи розшифровки трафіку на рівні проксі-серверів або інші технології для

отримання доступу до вмісту зашифрованих повідомлень. Проте ці методи також мають свої недоліки, зокрема збільшення навантаження на мережеву інфраструктуру та можливість зниження рівня конфіденційності даних.

7. Нестача кваліфікованих фахівців. Зростання складності сучасних мереж і кількості кіберзагроз вимагає наявності кваліфікованих фахівців, здатних не тільки налаштовувати системи моніторингу, але й реагувати на загрози. Проте на ринку праці існує дефіцит фахівців у галузі кібербезпеки, що ускладнює завдання для багатьох компаній.

Без достатньої кількості кваліфікованих кадрів компанії не можуть ефективно впроваджувати та підтримувати системи моніторингу, що призводить до підвищення ризику виникнення атак. Ця проблема потребує вирішення шляхом інвестування в навчання та підвищення кваліфікації працівників, а також впровадження автоматизованих систем моніторингу, які можуть зменшити залежність від людських ресурсів.

1.4. Критерії ефективності систем мережевої безпеки

Системи мережевої безпеки відіграють вирішальну роль у захисті інформаційних ресурсів від кібератак і зовнішніх загроз. Ефективність таких систем визначається їх здатністю своєчасно виявляти та нейтралізувати загрози, забезпечуючи стабільну та безперебійну роботу інформаційної інфраструктури. Однак для того, щоб оцінити, наскільки ефективно працює система мережевої безпеки, необхідно встановити конкретні критерії, що дозволять виміряти та порівняти результати.

Існує кілька основних критеріїв, які можуть використовуватися для оцінки ефективності систем мережевої безпеки. Ці критерії включають точність виявлення загроз, швидкість реагування, масштабованість, стійкість до атак, зручність адміністрування та економічну ефективність.

1. Точність виявлення загроз

Точність виявлення загроз є одним з найважливіших критеріїв оцінки ефективності систем мережевої безпеки. Цей критерій відображає здатність системи ідентифікувати реальні загрози і відрізняти їх від безпечної активності. Точність виявлення загроз може бути виміряна через два основні показники:

– Чутливість (*sensitivity*) – здатність системи виявляти всі загрози, включаючи нові та невідомі типи атак. Висока чутливість означає, що система фіксує більшість потенційних загроз, навіть якщо вони є незначними;

– Специфічність (*specificity*) – здатність системи мінімізувати кількість хибнопозитивних спрацювань. Висока специфічність забезпечує, що легітимна активність не позначається як загроза.

Оптимальна система мережевої безпеки повинна мати високі показники чутливості та специфічності, забезпечуючи виявлення загроз без надмірної кількості помилкових спрацювань.

2. Швидкість виявлення та реагування

Швидкість виявлення та реагування на загрози також є критично важливим показником. Чим швидше система може виявити загрозу, тим більше шансів запобігти її реалізації та мінімізувати наслідки. У контексті мережевої безпеки швидкість виявлення означає, що система здатна фіксувати аномалії або підозрілі дії у реальному часі або з мінімальною затримкою.

Швидкість реагування на загрозу визначає, як швидко система може заблокувати атаку або ізолювати компрометовані частини мережі для мінімізації шкоди. Відмінною рисою ефективних систем є можливість автоматичного реагування на загрози, що дозволяє уникнути ручних дій з боку фахівців з безпеки, тим самим зменшуючи час між виявленням та запобіганням атаки.

3. Масштабованість

Масштабованість – це здатність системи мережевої безпеки ефективно функціонувати у великих і складних мережах, що розширюються. У зв'язку з тим, що кількість пристроїв, зокрема в контексті Інтернету речей (*IoT*), постійно зростає, система безпеки повинна мати можливість адаптуватися до збільшення обсягів трафіку і кількості мережевих вузлів без втрати продуктивності.

Масштабованість також означає, що система повинна забезпечувати однаковий рівень захисту для різних частин мережі, незалежно від її розміру або складності. Ефективні системи безпеки дозволяють легко інтегрувати нові ресурси, мережеві компоненти та пристрої без необхідності суттєвого перепроектування існуючої інфраструктури.

4. Стійкість до атак та відмов

Стійкість системи до атак – це її здатність залишатися працездатною навіть у випадку серйозних кібератак. Окрім виявлення загроз, система безпеки повинна мати можливість продовжувати працювати у разі атак, що впливають на її компоненти або мережу в цілому. Це може включати наявність резервних каналів зв'язку, відмовостійких серверів, а також механізмів самовідновлення.

Стійкість до відмов означає, що система не повинна виходити з ладу під час інцидентів, і навіть якщо деякі її компоненти виходять з ладу, загальний рівень захисту залишається високим. Надійні системи безпеки включають функції автоматичного відновлення, такі як автоматичний перезапуск сервісів, резервування мережевих ресурсів або використання географічно розподілених центрів обробки даних.

5. Зручність адміністрування та керування

Системи мережевої безпеки повинні бути не тільки ефективними, але й зручними у використанні та адмініструванні. Цей критерій включає:

– Зручність налаштування: Система повинна легко налаштовуватись і конфігуруватись відповідно до вимог конкретної мережі;

– Можливість моніторингу: Фахівці з кібербезпеки повинні мати можливість отримувати повну інформацію про стан мережі, а також бути сповіщеними про потенційні загрози в режимі реального часу;

– Автоматизація: Сучасні системи безпеки повинні підтримувати автоматичне виявлення та реагування на загрози з мінімальним втручанням з боку адміністраторів.

Зручність використання також включає можливість інтеграції з іншими системами та засобами кібербезпеки, такими як фаєрволи, антивірусне програмне

забезпечення, системи виявлення та запобігання вторгнень (*IDS*/*IPS*) тощо. Інтуїтивно зрозумілий інтерфейс та інструменти для аналітики й звітності полегшують роботу адміністраторам безпеки.

6. Економічна ефективність

Економічна ефективність – це баланс між вартістю впровадження та підтримки системи безпеки і рівнем захисту, який вона забезпечує. Вартість системи включає як початкові витрати на впровадження (закупівля обладнання, ліцензії на програмне забезпечення), так і операційні витрати (підтримка, навчання персоналу, оновлення).

Ефективні системи безпеки повинні забезпечувати високий рівень захисту при оптимальному використанні ресурсів. Інвестування в систему кібербезпеки, яка може автоматично виявляти та реагувати на загрози, може бути дорожчим у початковій фазі, але в довгостроковій перспективі це може призвести до зниження операційних витрат завдяки зменшенню кількості хибнопозитивних спрацювань та потреби у людських ресурсах.

7. Інтеграція з іншими системами

Ефективні системи мережевої безпеки повинні бути інтегрованими з іншими інструментами та системами кібербезпеки, щоб забезпечити комплексний підхід до захисту. Це включає інтеграцію з фаєрволами, системами аутентифікації, антивірусами, системами збирання та аналізу журналів подій тощо.

Інтеграція дозволяє знизити складність керування безпекою та забезпечує єдине джерело даних для моніторингу і виявлення загроз. Вона також спрощує процес розслідування інцидентів безпеки, оскільки вся інформація доступна в одному місці.

1.5. Висновки до розділу 1

Виявлення вторгнень у комп'ютерні системи ґрунтується на поєднанні кількох підходів, що дозволяє ефективніше реагувати на різні типи загроз.

Сигнатурні методи добре працюють для виявлення відомих атак, поведінковий аналіз допомагає виявляти нові загрози, а кореляція подій і машинне навчання дають змогу виявляти складні атаки. Однак для забезпечення ефективного захисту необхідно використовувати ці методи комплексно, а також постійно оновлювати моделі та алгоритми виявлення, щоб пристосовуватися до нових викликів у сфері кібербезпеки.

Під час аналізу виявлено, що основними проблемами є складність сучасних мереж, обробка великих обсягів даних, висока кількість хибнопозитивних спрацювань, динамічність загроз, шифрування трафіку та захист розподілених мереж. Подолання цих проблем вимагає впровадження інноваційних технологій, таких як машинне навчання, розподілені системи моніторингу та спеціалізовані рішення для *IoT*-пристроїв.

Критерії ефективності систем мережевої безпеки допомагають оцінити надійність, продуктивність і економічну доцільність тих чи інших рішень у сфері кібербезпеки. Для забезпечення належного рівня захисту комп'ютерних мереж важливо, щоб система безпеки була точно налаштованою на виявлення загроз, швидко реагувала на атаки, масштабувалася разом із мережею, мала високу стійкість до збоїв та була зручною для адміністрування.

Оцінювання таких критеріїв дозволяє організаціям вибирати найбільш відповідні рішення для їхніх потреб, забезпечуючи як короткострокову, так і довгострокову ефективність впровадженої системи. Крім того, інтеграція сучасних методів, таких як штучний інтелект, автоматизовані системи реагування та хмарні технології, сприяє створенню комплексного підходу до безпеки, який здатний відповідати на нові виклики у сфері кіберзагроз.

Належним чином реалізовані системи мережевої безпеки дозволяють не лише запобігати атакам, але й забезпечувати безперервність бізнес-процесів, зменшувати фінансові ризики, а також підвищувати довіру клієнтів до організації. Таким чином, інвестування в ефективні системи захисту є не лише вимогою часу, а й важливим стратегічним кроком у напрямі цифрової трансформації.

Критерії ефективності мережевих систем безпеки повинні враховувати не лише технічні показники, а й адаптивність до мінливих умов сучасного кіберпростору. Це передбачає здатність системи навчатися новим загрозам, автоматично оновлювати бази даних сигнатур атак і впроваджувати поведінкові моделі для виявлення аномалій у трафіку. Такі можливості дозволяють значно знизити час виявлення та реакції на інциденти, що є критично важливим для мінімізації шкоди. Інтеграція з іншими системами моніторингу та управління мережею забезпечує синергію між різними компонентами захисту, підвищуючи загальну стійкість мережі.

Важливо також враховувати економічну складову впровадження та експлуатації систем мережевої безпеки. Надійне рішення має бути не лише технологічно ефективним, але й економічно обґрунтованим, забезпечуючи максимальну користь при мінімальних витратах. Це включає оптимізацію витрат на обладнання, програмне забезпечення, технічну підтримку та навчання персоналу. Впровадження таких систем є інвестицією у стійкість та конкурентоспроможність організації, що дозволяє ефективно функціонувати навіть у умовах підвищеного ризику кіберзагроз.

2.1. Методи дослідження мережевого трафіку для виявлення та блокування зовнішнього вторгнення

Методи дослідження мережевого трафіку спрямовані на аналіз потоків даних, виявлення потенційно небезпечних аномалій та їх блокування. Ці методи забезпечують комплексний підхід до моніторингу, аналізу та захисту мережі. Вони поділяються на кілька основних категорій, кожна з яких має свої підходи та інструменти.

2.1.1. Методи на базі сигнатурного аналізу

Сигнатурний аналіз є одним із найпоширеніших методів виявлення мережевих загроз. Його основний принцип полягає у використанні баз даних сигнатур, які зберігають шаблони відомих атак. Кожна сигнатура відповідає певному типу загрози, наприклад, експлойту, вірусу або шкідливій активності. Система порівнює вхідний мережевий трафік із цими сигнатурами, виявляючи збіги, які свідчать про можливу загрозу [12].

Бази сигнатур – це набори унікальних шаблонів, які характеризують певні види атак. Ці шаблони містять інформацію про:

– Комбінації команд, які використовуються у відомих шкідливих програмах;

– Структури пакетів, характерні для атак (наприклад, неправильні або підроблені заголовки, унікальні *payload*-и);

– Поведінкові патерни, наприклад, послідовність запитів у протоколах *HTTP*, *FTP*, *DNS* тощо.

Приклади використання

1. *Snort* – це популярна система виявлення вторгнень, яка використовує сигнатурний аналіз. База сигнатур *Snort* містить тисячі правил для виявлення загроз, включаючи мережеві атаки, експлойти, сканування портів тощо.

Приклад правила *Snort*:

alert tcp any any -> 192.168.1.0/24 80 (*content*:"/*etc*/*passwd*"; *msg*:"*Possible Password File Access*"; *sid*:1001;)

Це правило генерує сповіщення, якщо в *HTTP*-запиті міститься доступ до файлу /*etc*/*passwd*.

2. *Suricata* – це потужна система виявлення вторгнень, яка підтримує багатопоточну обробку даних і працює з базами сигнатур, сумісними зі *Snort*. *Suricata* дозволяє використовувати сигнатурний аналіз разом із поведінковим.

Принцип дії:

1. Система захоплює мережевий трафік.

2. Для кожного пакета виконується порівняння з шаблонами в базі сигнатур.

3. Якщо трафік відповідає шаблону, генерується сигналізація або виконується блокування.

Етапи обробки порівняння:

Етап 1: Захоплення трафіку:

– *IDS*/*IPS* системи використовують пакети, захоплені в реальному часі, або агреговані потоки даних;

– Пакет аналізується на рівні заголовків (*IP*, *TCP*, *UDP*) і вмісту (*payload*).

Етап 2: Аналіз на рівні сигнатур:

– Кожен пакет порівнюється з шаблонами з бази сигнатур;

– Наприклад, якщо заголовок пакету містить *IP*-адресу або порт, які відповідають відомій атаці (наприклад, скануванню портів або атаці типу *DoS*), система вважає його підозрілим.

Етап 3: Генерація сигналів:

– У разі збігу система генерує сигнал тривоги, який може включати: інформування адміністратора, автоматичне блокування пакету або джерела трафіку (у разі використання *IPS*).

Приклад застосування

– *HTTP*-запити: якщо шаблон сигнатури містить фрагмент '/*login.php*?*user*=', а запит з мережі відповідає цьому шаблону, система може визначити спробу *SQL*-ін'єкції;

– *ICMP*-трафік: для атаки типу *Ping Flood* шаблон може включати перевищення певної частоти запитів *ICMP Echo Request*.

Основні проблеми сигнатурного аналізу:

1. Нездатність виявляти нові або модифіковані атаки:

– Сигнатурний аналіз ефективний лише для відомих атак;

– Нові загрози, які не мають шаблону в базі сигнатур (атаки "нульового дня"), залишаються непоміченими;

– Наприклад, якщо зловмисник змінює команди в експлойті, сигнатура стає непридатною.

2. Потреба в постійному оновленні бази:

– База сигнатур повинна оновлюватися регулярно для включення нових шаблонів атак;

– Застарілі бази неефективні проти сучасних загроз.

3. Високий рівень ресурсоспоживання:

– Порівняння кожного пакета з великою базою сигнатур може вимагати значних обчислювальних ресурсів, особливо в мережах із великим обсягом трафіку.

4. Обхід сигнатурного аналізу

– Зловмисники можуть модифікувати свої атаки, щоб обійти сигнатурний аналіз:

– Використання шифрування для приховування вмісту трафіку;

– Розбиття шкідливих команд на кілька частин, щоб уникнути збігу із шаблоном.

Сигнатурний аналіз слід комбінувати з іншими методами, наприклад, поведінковим аналізом або машинним навчанням, щоб забезпечити комплексний підхід до виявлення загроз [13].

Сигнатурний аналіз є важливим інструментом в арсеналі систем виявлення вторгнень. Він дозволяє ефективно виявляти відомі загрози завдяки використанню баз сигнатур і порівнянню трафіку з шаблонами атак. Однак для забезпечення високого рівня безпеки необхідно долати його обмеження шляхом інтеграції з іншими методами аналізу та регулярного оновлення бази даних сигнатур.

2.1.2. Методи на базі поведінкового аналізу

Методи поведінкового аналізу (*Behavioral Analysis*) базуються на принципі визначення відхилень у поведінці мережевого трафіку від нормальних шаблонів, які є типовими для системи або мережі. Відмінність поведінкового аналізу від сигнатурного підходу полягає в тому, що він не обмежується базами відомих атак, а намагається виявити загрози, аналізуючи зміни у поведінці трафіку, навіть якщо атака не була раніше зафіксована.

Цей підхід особливо ефективний для виявлення:

– Атак типу нульового дня (*Zero-Day Attacks*);

– Продвинутих постійних загроз (*Advanced Persistent Threats, APTs*);

– Внутрішніх загроз (*Insider Threats*), коли аномальна поведінка походить від легітимних користувачів.

Ключові принципи поведінкового аналізу

1. Аналіз історичних даних:

– Вивчення поведінки користувачів, пристроїв і загальної мережевої активності протягом тривалого періоду часу;

– Побудова статистичних моделей нормальної поведінки.

2. Порівняння поточної активності з моделями:

– Поточний трафік аналізується у реальному часі, і будь-яке відхилення від моделей визначається як потенційна загроза.

3. Адаптивність:

– Моделі оновлюються відповідно до змін у поведінці мережі, що дозволяє уникати застарілості та фіксувати нові шаблони.

Нормальна модель – це статистична або математична репрезентація поведінки мережі у звичайному стані. Створення таких моделей є фундаментом для поведінкового аналізу.

1. Джерела даних для моделі:

– Мережевий трафік: Обсяги передачі даних, порти, *IP*-адреси, протоколи, тривалість сесій;

– Логи системи: Вхід користувачів, запуск додатків, зміна конфігурацій;

– Активність користувачів: Типові дії, доступ до файлів, частота запитів до серверів.

2. Методи створення моделей:

А) Статистичні підходи:

– Використання середніх значень, стандартних відхилень, інтерквартильних діапазонів;

– Наприклад, середній обсяг *HTTP*-трафіку на сервер – 100 МБ/година.

Б) Графові моделі:

– Вивчення взаємодій між вузлами мережі у вигляді графів;

– Виявлення аномальних зв’язків між вузлами.

В) Моделі часових рядів:

– Аналіз даних, що змінюються у часі, наприклад, кількість з'єднань за хвилину.

3. Приклади нормальної поведінки:

– Користувач А отримує доступ до внутрішнього сервера тільки з 8:00 до 18:00;

– Сервер Б обробляє запити від 20 *IP*-адрес у середньому з частотою 50 запитів/хвилину.

Аномалії – це відхилення від створених нормальних моделей. Їх виявлення є ключовим завданням поведінкового аналізу.

Типи аномалій:

1. Об’ємні аномалії:

– Незвично великий або маленький обсяг трафіку;

– Наприклад, обсяг вихідного трафіку з сервера раптово виріс на 300% без явної причини.

2. Аномалії частоти: підозріла частота запитів до сервера. Наприклад, один користувач почав надсилати сотні запитів за хвилину, хоча зазвичай він виконує лише 10.

3. Аномалії поведінки:

– Зміни у типовій активності користувача;

– Наприклад, звичайний користувач раптово починає запускати адміністраторські команди.

Методи виявлення аномалій:

1. Пороги (*Thresholds*):

– Встановлення фіксованих меж для різних параметрів трафіку;

– Приклад: якщо частота запитів перевищує 100/хвилину, це може бути атакою.

2. Евристичний аналіз:

– Використання правил для виявлення підозрілих моделей поведінки;

– Наприклад, виявлення спроби входу з незвичайної геолокації.

3. Статистичний аналіз: порівняння поточного значення із середнім або стандартним відхиленням.

4. Аналіз часових рядів: використання моделей *ARIMA* для визначення відхилень у частоті запитів у часі.

Приклади:

– Зловмисник здійснює атаку типу "брютфорс", під час якої відправляє сотні запитів до серверу для підбору паролю;

– Аномальна активність із незвичних *IP*-адрес може вказувати на ботнет.

Алгоритми машинного навчання забезпечують автоматизацію аналізу даних та адаптацію до нових загроз. Вони ефективно працюють із великими наборами даних та дозволяють виявляти складні аномалії, які важко ідентифікувати вручну.

Основні алгоритми машинного навчання:

1. Класифікація (*Classification*): використовується для категоризації трафіку на "нормальний" або "аномальний".

Найбільш розповсюджені алгоритми:

– *SVM* (*Support Vector Machine*): створює гіперплощину для розділення класів, ефективний для нелінійних даних;

– *Random Forest*: Використовує деревовидні моделі для прогнозування аномалій, висока точність і стійкість до "шуму" в даних.

2. Кластеризація (*Clustering*): виявляє групи подібних даних та аномалії, які не належать до жодного кластера.

Найбільш розповсюджені алгоритми:

– *K-Means*: групує схожі дані в кластери, аномалії визначаються як точки, які не відповідають жодному кластеру;

– *DBSCAN* (*Density-Based Spatial Clustering*): виявляє аномалії в малозаселених областях даних.

3. Глибоке навчання (*Deep Learning*): використовується для аналізу складних патернів у даних. Наприклад, нейронні мережі *LSTM* (*Long Short-Term Memory*) ефективно працюють із часовими рядами.

Етапи роботи алгоритмів:

1. Збір і попередня обробка даних: очищення даних, видалення шуму, нормалізація.

2. Навчання моделі: алгоритм навчається на даних, створюючи модель поведінки.

3. Прогнозування та виявлення аномалій: модель оцінює нові дані й ідентифікує відхилення. Модель *SVM* може навчитися визначати нормальний *HTTP*-трафік і позначати спроби *SQL*-ін'єкції як аномальні.

Переваги поведінкового аналізу

1. Здатність виявляти нові загрози: виявляє атаки, які не мають сигнатур.

2. Адаптивність: моделі оновлюються на основі змін у поведінці мережі.

3. Ефективність для складних атак: багатоступеневі атаки або *APT*.

Недоліки поведінкового аналізу:

1. Велика кількість хибнопозитивних спрацювань: будь-яке незвичне відхилення може помилково розцінюватися як загроза.

2. Ресурсомісткість: аналіз великих обсягів даних вимагає значних обчислювальних ресурсів.

3. Складність побудови моделей: необхідний тривалий період для збору історичних даних.

Методи поведінкового аналізу є невід'ємною частиною сучасних систем виявлення вторгнень, оскільки вони дозволяють виявляти як відомі, так і нові загрози. Використання нормальних моделей, аномалій та алгоритмів машинного навчання забезпечує високий рівень точності та адаптивності системи. Однак їх ефективність значною мірою залежить від якості даних, налаштувань моделі та обчислювальних потужностей.

2.1.3. Аналіз потоків даних

Аналіз потоків даних (*Flow Analysis*) є одним із найефективніших методів моніторингу мережевого трафіку для виявлення та запобігання загрозам. Основна ідея методу полягає у зборі та аналізі метаданих про мережеві потоки, таких як джерело, призначення, обсяг, тривалість та інші характеристики передачі даних, без аналізу вмісту самих пакетів. Цей підхід дозволяє забезпечити швидкий та ефективний моніторинг великих мережевих середовищ [14].

Потік даних – це послідовність пакетів, що передаються між двома кінцевими точками (*IP*-адресами та портами) в межах певного інтервалу часу. Потік визначається такими параметрами:

– *IP*-адреса джерела;

– *IP*-адреса призначення;

– Порт джерела;

– Порт призначення;

– Протокол передачі (*TCP, UDP, ICMP* тощо);

– Час початку та завершення передачі;

– Кількість переданих пакетів і байтів.

Основні етапи аналізу потоків:

1. Збір метаданих: дані про потоки збираються за допомогою спеціалізованих протоколів, таких як *NetFlow, sFlow, IPFIX*, або інструментів, таких як *Argus*.

2. Агрегація даних: метадані агрегуються для побудови потоків. Наприклад, усі пакети між одними й тими самими джерелом і призначенням об'єднуються в один потік.

3. Аналіз характеристик: оцінюються параметри потоків, такі як обсяг, тривалість, частота з'єднань.

4. Виявлення аномалій: виявляються підозрілі патерни, наприклад, надмірна кількість потоків до одного порту, тривалі сесії без передавання даних тощо.

Ключові аспекти аналізу потоків:

1. Методи збору потоків, де існує кілька протоколів і технологій для збору даних про потоки:

– *NetFlow*: розроблений *Cisco*. Збирає дані про потоки на маршрутизаторах та комутаторах. Дані включають *IP*-адреси, порти, протокол, кількість байтів і пакетів;

– *sFlow*: використовує вибірковий збір даних із мережі (*sampling*), що дозволяє працювати в умовах високого навантаження;

– *IPFIX* (*IP Flow Information Export*): стандартний протокол для експорту даних про потоки, що є наступником *NetFlow*;

– *Argus*: інструмент для аналізу потоків, який підтримує створення гнучких звітів про мережеві потоки.

2. Основні метрики для аналізу потоків

– Обсяг трафіку (*Bytes*): загальна кількість переданих байтів у потоці;

– Частота пакетів (*Packets*): кількість переданих пакетів у потоці;

– Тривалість (*Duration*): час існування потоку;

– Протокол (*Protocol*): *TCP, UDP, ICMP* тощо;

– Частота з'єднань (*Connections per second*): кількість потоків, ініційованих за одиницю часу.

3. Аналіз поведінки потоків, який може визначити наступні аномалії:

– раптове збільшення трафіку: може свідчити про *DDoS*-атаку;

– часте відкриття та закриття з'єднань: може бути результатом сканування портів;

– довгі сесії: трафік без активності може бути ознакою вторгнення або передачі даних.

Переваги аналізу потоків:

1. Мінімізація обсягу даних для аналізу: замість аналізу всього трафіку система працює з метаданими, що значно знижує обсяг оброблюваної інформації.

2. Швидкість обробки: аналіз метаданих є менш ресурсомістким, ніж глибокий аналіз пакетів (*DPI*), що робить його придатним для великих мереж.

3. Масштабованість: аналіз потоків може бути ефективно використаний у великих корпоративних мережах та хмарних середовищах.

4. Виявлення складних атак: аналіз потоків дозволяє ідентифікувати патерни поведінки, які характерні для багатоступеневих атак.

Обмеження аналізу потоків:

1. Відсутність аналізу вмісту пакетів: потоковий аналіз не дозволяє аналізувати *payload* (вміст) пакетів, тому він не може виявляти загрози, приховані у шифрованому трафіку.

2. Обмежена деталізація: потоки містять лише загальні характеристики (метадані), що може ускладнювати точну ідентифікацію загрози.

3. Вразливість до підроблених потоків: зловмисники можуть створювати потоки з нормальними характеристиками, що ускладнює їх виявлення.

4. Залежність від інфраструктури: ефективність аналізу потоків залежить від якості налаштування пристроїв збору та обробки даних (маршрутизатори, комутатори тощо).

Виявлення загроз за допомогою аналізу потоків:

1. Виявлення *DoS*/*DDoS* атак: аналіз потоків дозволяє ідентифікувати атаки, коли великий обсяг трафіку спрямовується до одного порту або *IP*-адреси.

Метрики:

– надмірна кількість потоків до одного вузла;

– великий обсяг *UDP*-трафіку.

2. Виявлення сканування портів: потоковий аналіз допомагає виявити спроби сканування мережі, коли з одного джерела надсилаються запити до різних портів.

Метрики:

– частота відкриття/закриття з'єднань;

– збільшення кількості потоків із одного джерела.

3. Виявлення аномалій у використанні протоколів: аналіз потоків допомагає визначити незвичні протоколи або поведінку [15]. Наприклад, несподівана активність *FTP*-протоколу у мережі, де він зазвичай не використовується.

4. Виявлення витоку даних: довготривалі потоки або потоки з великим обсягом вихідного трафіку можуть свідчити про витік даних.

Застосування аналізу потоків:

1. Корпоративні мережі: виявлення аномалій у трафіку між офісними пристроями та серверами.

2. Хмарні середовища: моніторинг потоків між віртуальними машинами для запобігання внутрішнім загрозам.

3. Критична інфраструктура: контроль мереж енергетики, транспорту та комунікацій.

Інструменти для аналізу потоків:

1. *NetFlow Analyzer*: інструмент для моніторингу та аналізу трафіку за допомогою *NetFlow*.

2. *Argus*: потужний інструмент для збору та аналізу потоків у реальному часі.

3. *ntopng*: візуалізація потоків і детальний аналіз у реальному часі.

Аналіз потоків даних є ефективним методом для виявлення загроз у великих мережах завдяки використанню метаданих замість аналізу всього вмісту пакетів. Він забезпечує масштабованість, швидкість обробки та здатність виявляти складні загрози, такі як *DDoS*-атаки, сканування портів і витоки даних. Хоча він має свої обмеження, наприклад, відсутність аналізу вмісту, інтеграція аналізу потоків із іншими методами, такими як поведінковий або сигнатурний аналіз, дозволяє створити комплексну систему захисту.

2.1.4. Глибокий аналіз пакетів (*Deep Packet Inspection, DPI*)

Глибокий аналіз пакетів (*DPI*) – це технологія моніторингу мережевого трафіку, яка дозволяє аналізувати не лише заголовки мережевих пакетів, але й їхній вміст [16]. На відміну від базового аналізу, який обмежується параметрами *IP*-адрес, портів і протоколів, *DPI* забезпечує детальне вивчення *payload* (корисного навантаження) пакета, що дає змогу виявляти складні загрози, приховані в даних, і забезпечувати високий рівень безпеки.

Принцип роботи *DPI*:

1. Перехоплення трафіку: *DPI* отримує доступ до всього мережевого трафіку, що проходить через пристрій (маршрутизатор, фаєрвол або сервер моніторингу).

2. Аналіз заголовків: аналізуються стандартні поля заголовків пакета (*IP*-адреси, порти, прапорці *TCP* тощо), щоб визначити базову інформацію про маршрут передачі даних.

3. Аналіз *payload* (вмісту): здійснюється перевірка вмісту пакета, включаючи текстові дані, файли, мультимедійний контент та команди, що передаються між клієнтом і сервером.

4. Ідентифікація патернів: *DPI* порівнює вміст пакета з відомими шаблонами шкідливого програмного забезпечення, сигнатурами атак чи іншими маркерами загроз.

5. Реакція: у разі виявлення загрози *DPI* може блокувати пакет, позначати його як підозрілий або надсилати сигнал тривоги.

Особливості та функціональність *DPI*:

1. Аналіз на всіх рівнях мережевої моделі (*OSI*): *DPI* працює на рівнях від 2-го (канального) до 7-го (прикладного) рівня моделі *OSI*, що дозволяє отримати детальну інформацію про трафік. Приклад: на 3-му рівні (мережевий): аналіз *IP*-адрес і маршрутів, на 4-му рівні (транспортний): аналіз *TCP*/*UDP*-з'єднань, на 7-му рівні (прикладний): виявлення специфічних *HTTP*, *DNS*, *FTP* запитів.

2. Виявлення шкідливого вмісту: *SQL*-ін'єкції (команди в *payload*, які використовують уразливості баз даних), *XSS* (*Cross-Site Scripting*) небезпечний *JavaScript*-код у веб-запитах, *Phishing*: шкідливі *URL* у трафіку.

3. Аналіз зашифрованого трафіку: *DPI* може працювати із зашифрованим трафіком за допомогою технологій *SSL*/*TLS*-інспекції, що дозволяє вивчати вміст *HTTPS*-запитів.

4. Класифікація та фільтрація трафіку: *DPI* може визначати типи передаваних даних (веб-сторінки, відео, файли тощо) і застосовувати політики фільтрації.

5. Генерація детальних звітів: системи *DPI* надають адміністратору детальні звіти про мережевий трафік, включаючи: найбільш використовувані *IP*-адреси, протоколи, що використовуються, потоки з підозрілим вмістом [17].

Методи реалізації *DPI*:

1. Використання сигнатурного аналізу: *Payload* пакета порівнюється із базою сигнатур, яка містить шаблони відомих загроз. Приклад: система визначає шкідливий код у *HTTP*-запиті, який відповідає сигнатурі *SQL*-ін'єкції.

2. Поведінковий аналіз: аналізуються аномалії у поведінці трафіку. Наприклад, підозрілі з'єднання до нетипових портів або передачі великих обсягів даних до невідомих адрес.

3. Контентний аналіз: аналізується на наявність небажаного контенту (неприпустимі файли (великі вкладення в електронній пошті), небезпечний код у тексті).

4. *SSL*/*TLS*-інспекція: *DPI* розшифровує зашифрований трафік за допомогою *MITM*-підходу (*Man-In-The-Middle*) або інших методів: використання спеціальних сертифікатів для перехоплення *HTTPS*-трафіку, декодування та аналіз перед повторною передачею.

Глибокий аналіз пакетів (*DPI*) є потужним методом забезпечення мережевої безпеки, який дозволяє детально аналізувати трафік на всіх рівнях. Цей метод забезпечує високий рівень точності виявлення загроз, але вимагає значних ресурсів для реалізації та може порушувати конфіденційність. *DPI* найефективніший у поєднанні з іншими методами, такими як сигнатурний або поведінковий аналіз, для створення багаторівневої системи захисту.

2.1.5. Аналіз журналів (*Log Analysis*)

Аналіз журналів (*Log Analysis*) – це метод вивчення подій, записаних у лог-файли системи, мережевих пристроїв та додатків для виявлення аномалій, загроз або вторгнень. Логи містять інформацію про всі дії, що відбувалися в системі або мережі, такі як спроби входу, зміну конфігурації, виклики до *API*, звернення до веб-серверів тощо [18]. Цей метод дозволяє оцінювати безпеку системи, виявляти причини проблем і запобігати майбутнім атакам.

Основні джерела даних для аналізу журналів

1. Журнали операційних систем (*System Logs*): інформація про системні події: завантаження, збої, спроби входу та виходу користувачів:

– /*var*/*log*/*auth.log* у *Linux* містить інформацію про спроби аутентифікації;

– *Windows Event Viewer* фіксує події системи.

2. Журнали додатків (*Application Logs*): логи веб-серверів (*Apache, Nginx*), баз даних (*MySQL, PostgreSQL*), поштових серверів тощо. Містять інформацію про запити до серверів, *SQL*-запити, помилки додатків.

3. Мережеві журнали (*Network Logs*): інформація від маршрутизаторів, комутаторів, фаєрволів, *VPN*. Фіксують *IP*-адреси джерел і призначень, порти, протоколи.

4. Системи виявлення вторгнень (*IDS/IPS Logs*): журнали систем типу *Snort, Suricata,* які містять інформацію про виявлені загрози та спроби вторгнень.

5. Системи моніторингу (*Monitoring Systems*): дані від інструментів типу *Nagios, Zabbix, Prometheus,* що відображають стан систем.

6. Журнали хмарних сервісів: логи доступу, подій, *API*-запитів у хмарних середовищах (*AWS CloudWatch, Azure Monitor*).

Основні етапи аналізу журналів:

1. Збір журналів: журнали збираються з різних джерел, централізуються та зберігаються в одному місці для подальшого аналізу (*Syslog*: стандартний протокол для збору логів з різних пристроїв, *Logstash*: частина *ELK*-стеку, яка агрегує журнали з багатьох джерел, *Fluentd*: інструмент для об'єднання логів у реальному часі);

2. Попередня обробка: виконується нормалізація логів (видалення зайвих даних, дублюючих записів, перетворення даних у єдиний формат);

3. Аналіз даних: використовуються різні підходи для вивчення логів (сигнатурний аналіз: пошук в логах записів, які відповідають відомим шаблонам атак, аналіз трендів: оцінка змін у логах за певний період часу, поведінковий аналіз: виявлення відхилень від нормальної поведінки);

4. Кореляція подій: кореляція даних із різних джерел дозволяє виявляти складні атаки (зв'язок між записами про сканування портів у журналах фаєрволу та спробами входу в систему);

5. Візуалізація результатів: візуалізація допомагає адміністратору зрозуміти ключові події та тенденції (*Kibana*: створення графіків і дашбордів для аналізу логів, *Grafana*: візуалізація трендів у реальному часі).

Методи аналізу журналів [19]:

1. Сигнатурний аналіз: виявлення подій на основі шаблонів, які відповідають відомим атакам. Приклад: пошук у логах веб-сервера записів із підозрілими запитами: *GET /login.php?user=' OR* '1'='1'

2. Поведінковий аналіз: аналізує логи для виявлення аномалій у поведінці системи чи користувачів. Приклад: підозрілі спроби входу з нових *IP*-адрес або з нестандартного географічного місця.

3. Кореляційний аналіз: встановлення взаємозв'язків між подіями. Приклад: аналіз логів *IDS* і фаєрволу для виявлення зв'язку між скануванням портів і спробами входу.

4. Машинне навчання: використання алгоритмів для автоматичного виявлення аномалій. Алгоритми: кластеризація (групування схожих подій і виявлення тих, що відрізняються), класифікація (ідентифікація подій як "нормальних" чи "аномальних").

5. Темпоральний аналіз: аналіз змін у логах за часом. Приклад: підвищена активність запитів до сервера вночі може свідчити про автоматизовану атаку.

Виявлення загроз за допомогою аналізу журналів:

1. Виявлення спроб входу (пошук у логах системи записів про несанкціоновані спроби входу. Приклад: часті помилки аутентифікації (*Failed password for user*);

2. Виявлення змін у конфігурації (логи зберігають інформацію про зміни в системних файлах або налаштуваннях. Приклад: запис про зміну файлу /*etc*/*passwd* може свідчити про несанкціоновані дії);

3. Виявлення витоку даних (аналіз логів для ідентифікації незвичайної активності, пов'язаної з передачею великих обсягів даних);

4. Виявлення мережевих атак (логи фаєрволів і *IDS* дозволяють виявляти атаки, такі як сканування портів або *DoS*-атаки).

Інструменти для аналізу журналів

1. *Splunk*: потужний комерційний інструмент для збору, обробки та аналізу логів.

2. *ELK Stack* (*Elasticsearch, Logstash, Kibana*): відкрите програмне забезпечення для збору, обробки та візуалізації логів.

3. *Graylog*: інструмент із функціями аналізу, пошуку та зберігання логів.

4. *SIEM*-системи (*Security Information and Event Management*): *IBM QRadar, ArcSight,* які дозволяють корелювати події з різних джерел.

Переваги аналізу журналів:

1. Висока деталізація: логи надають детальну інформацію про кожну подію.

2. Історичний аналіз: можливість вивчення подій, що сталися в минулому.

3. Кореляція подій: інтеграція даних із різних джерел дозволяє виявляти складні атаки.

4. Моніторинг у реальному часі: використання інструментів для збору та аналізу логів у реальному часі.

Недоліки аналізу журналів

1. Великий обсяг даних: великі організації генерують терабайти логів, що ускладнює їх обробку.

2. Потреба в автоматизації: ручний аналіз логів є неефективним для великих середовищ.

3. Складність кореляції: встановлення зв’язків між подіями з різних джерел може бути технічно складним.

Аналіз журналів є одним із найпотужніших методів для забезпечення безпеки мереж та систем, що дозволяє виявляти спроби вторгнення, аномальну активність і потенційні загрози на основі історичних і поточних даних.

2.2. Методи блокування зовнішнього вторгнення в комп'ютерні системи

Блокування пакетів з ознаками загроз на мережевому рівні виконується за допомогою спеціальних механізмів, що можуть бути реалізовані як на обладнанні, так і в програмних рішеннях. Для здійснення блокування зазвичай використовуються фаєрволи, маршрутизатори з функціями безпеки, системи виявлення та запобігання вторгнень (*IPS*), які аналізують пакети на основі певних правил.

1. Створення правил блокування на основі сигнатур [5]:

– Етап 1: створюється базу відомих сигнатур атак. Це може бути, наприклад, шаблон або специфічний паттерн, який використовує відомий шкідливий трафік. Бази сигнатур часто поставляються з програмним забезпеченням *IDS*/*IPS* (*Snort*, *Suricata*);

– Етап 2: налаштовується фаєрвол або *IPS* для аналізу заголовків і вмісту кожного пакета, що проходить через систему. Якщо пакет відповідає певній сигнатурі загрози, фаєрвол блокує цей пакет;

– Етап 3: перевірка, що правила блокування активуються автоматично. Як тільки пакет відповідає сигнатурі загрози, він має бути заблокований без втручання адміністратора.

2. Аналіз аномалій для блокування [5]:

– Етап 1: визначаються модель нормальної поведінки мережі. Це може включати частоту з’єднань, типи протоколів, використання певних портів;

– Етап 2: налаштовується система моніторингу для відстеження аномалій. Це може бути зміна кількості з'єднань або спроби підключення до незвичайних портів;

– Етап 3: реалізовуються правила, що блокують пакети, які виходять за рамки цих норм. Наприклад, встановлюється поріг частоти підключень на один *IP*-адрес або кількість запитів на сервер.

3. Блокування на основі заголовків протоколів

– Етап 1: налаштовується аналізатор пакетів на перевірку заголовків *IP* і *TCP*-пакетів. Фільтруйте за конкретними параметрами, такими як *IP*-адреса джерела, порт призначення або прапорці *TCP*;

– Етап 2: створюється правило для блокування всіх пакетів, що надходять з підозрілих *IP*-адрес або використовують заборонені порти (наприклад, сканування портів);

– Етап 3: додається логіка для автоматичного блокування або відхилення пакетів із неправильними або маніпульованими заголовками (наприклад, пакети з підробленими *IP*-адресами або неправильно встановленими прапорцями *TCP*).

Також існують методи блокування на основі політик доступу (правил, які визначають, хто, коли і з яких пристроїв може підключатися до мережі або до конкретних ресурсів системи). Механізми блокування на основі політик доступу забезпечують контроль трафіку відповідно до заздалегідь визначених правил [20].

1. Створення політик на основі *IP*-адрес

– Етап 1: визначаються допустимі *IP*-адреси для доступу до певних сегментів мережі або сервісів. Це може бути список "білих" або "чорних" адрес;

– Етап 2: налаштовується фаєрвол або маршрутизатор для блокування всіх пакетів, що надходять від *IP*-адрес, які не входять до дозволеного списку. Можна використовувати *Access Control Lists* (*ACLs*);

– Етап 3: створюється політики для різних мережевих сегментів або серверів. Наприклад, певні *IP*-адреси можуть мати доступ до внутрішніх серверів, але не до серверів баз даних.

2. Політики на основі часу доступу

– Етап 1: визначаються часові інтервали, коли доступ до мережевих ресурсів має бути дозволений або заборонений для певних користувачів або пристроїв;

– Етап 2: реалізовується блокування трафіку на основі цих часових політик. Наприклад, доступ до внутрішньої мережі може бути дозволений тільки в робочі години;

– Етап 3: використовується механізми в фаєрволах або системах управління доступом для автоматичного застосування цих правил.

3. Блокування на основі типу трафіку

– Етап 1: налаштовується систему для аналізу типу передаваних даних або протоколів. Наприклад, дозволяйте *HTTP*-трафік, але блокуйте небезпечний протокол *FTP*;

– Етап 2: створюється політики доступу для різних типів даних. Забороняйте передачу файлів певного типу або обмежуйте трафік до певних протоколів;

– Етап 3: забезпечується динамічне блокування або редагування політик для нових типів трафіку або протоколів, що з'являються.

Для протидії *DoS*/*DDoS*-атакам, що спрямовані на перевантаження серверів або мережевої інфраструктури великою кількістю запитів, використовують різні методи блокування та управління трафіком:

1. Фільтрація трафіку на основі *IP*-адрес:

– Етап 1: виявляються *IP*-адреси джерел, з яких здійснюється масове надходження запитів під час атаки;

– Етап 2: створюється правила в фаєрволі або на рівні маршрутизатора для блокування запитів із цих адрес. Ви також можете використовувати автоматизовані рішення, що працюють у реальному часі для швидкого фільтрування підозрілого трафіку;

– Етап 3: під час *DDoS*-атак часто використовують ботнети. Реалізовується механізми для автоматичного блокування великих груп підозрілих адрес або використовується геофільтрацію для блокування певних регіонів.

2. Розподіл навантаження (*Load Balancing*):

– Етап 1: налаштовується мережеву інфраструктуру для розподілу вхідних запитів на кілька серверів або центрів обробки даних. Це допоможе уникнути перевантаження окремих серверів під час атаки;

– Етап 2: використовується механізми для відсіву "шкідливого" трафіку, перенаправляючи його на сервери, що спеціально призначені для відхилення атаки;

– Етап 3: реалізовується динамічне масштабування серверів. Якщо відбувається раптове збільшення навантаження, додайте додаткові сервери для обробки легітимного трафіку.

3. Обмеження швидкості (*Rate Limiting*):

– Етап 1: встановлюється поріг для кількості запитів від одного джерела (*IP*-адреси) за певний період часу. Це обмежить можливість перевантаження серверів великою кількістю запитів;

– Етап 2: реалізовується механізми для динамічного регулювання цього порогу. Під час атаки поріг можна зменшити, щоб обмежити кількість запитів від потенційних атакуючих;

– Етап 3: використовується алгоритми для ідентифікації нормальної поведінки користувачів і відокремлення шкідливого трафіку від легітимного. Це дозволить знизити негативний вплив на легітимних користувачів під час атаки.

2.3. Аналіз роботи наявних систем моніторингу мережевого трафіку

Для оцінки популярних засобів моніторингу мережевого трафіку, які відіграють ключову роль у забезпеченні інформаційної та кібербезпеки, наведемо аналіз їхніх основних можливостей та функцій. *Wireshark* є одним із найвідоміших інструментів для аналізу мережевого трафіку, що забезпечує перехоплення та детальний аналіз пакетів у реальному часі. Завдяки цьому інструменту можна виявляти потенційні загрози та детально досліджувати структуру трафіку. *Nagios* спеціалізується на моніторингу стану мережевих пристроїв, серверів та програмного забезпечення, пропонуючи функції виявлення аномалій і оперативного сповіщення про проблеми. *PRTG Network Monitor* дозволяє отримувати звіти про стан мережі, включаючи швидкість передачі даних, стан пристроїв та аналіз трафіку в реальному часі, що робить його універсальним рішенням для моніторингу інфраструктури. *SolarWinds Network Performance Monitor* забезпечує комплексний підхід до моніторингу мережі, зокрема детальний аналіз трафіку, ідентифікацію проблем та оптимізацію роботи мережевої інфраструктури, що особливо корисно для великих корпоративних мереж. Нарешті, *Splunk* пропонує потужну платформу для збору, аналізу та обробки даних з різних джерел, що включає виявлення аномалій у трафіку та реагування на кіберзагрози в режимі реального часу. Кожен із цих інструментів має свої унікальні особливості, які роблять їх корисними у різних сценаріях забезпечення безпеки та моніторингу мереж.

Wireshark – це один із найвідоміших та найпотужніших інструментів для аналізу мережевого трафіку. Основна перевага *Wireshark* полягає в його здатності працювати в реальному часі, перехоплюючи пакети даних, що проходять через мережу, і надаючи докладну інформацію про кожен з них (рис. 2.1).

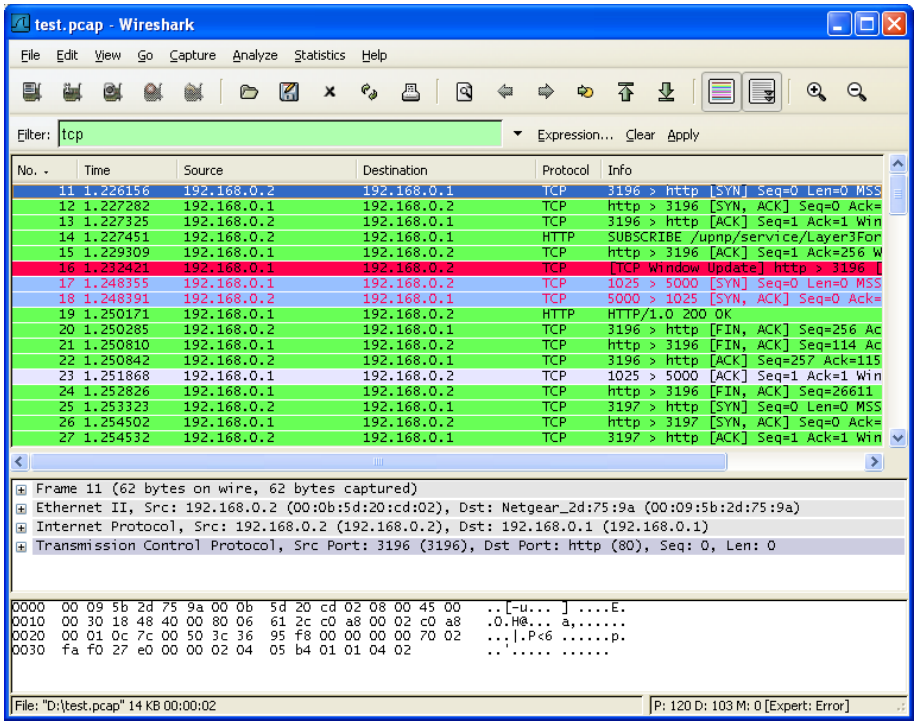

Рис. 2.1. Приклад діаграми розподілу трафіку, який отримано програмою *Wireshark*

Можливості *Wireshark*:

– перехоплення та аналіз трафіку на всіх рівнях моделі *OSI*;

– декодування понад 1,000 різних протоколів, включаючи *HTTP*, *FTP*, *TCP*, *UDP*, *DNS* тощо;

– глибокий аналіз пакетів з відображенням заголовків та корисного навантаження;

– фільтрування трафіку за заданими критеріями (*IP*-адреса, порт, тип протоколу тощо);

– аналіз уразливостей у мережі, наприклад, виявлення незашифрованих даних.

Переваги:

– безкоштовне програмне забезпечення з відкритим кодом;

– велика база навчальних матеріалів і спільнота користувачів;

– доступність для різних операційних систем (*Windows, macOS, Linux*).

Недоліки:

– високий рівень технічної складності, що ускладнює використання для початківців;

– великий обсяг зібраних даних може бути складним для аналізу без попередньої фільтрації.

Snort – система виявлення вторгнень у мережу за допомогою сигнатурного аналізу.

Особливості функціоналу:

– використовує базу сигнатур для швидкого розпізнавання відомих атак;

– може працювати в режимах *IDS* (виявлення) або *IPS* (запобігання);

– підтримує фільтрування, аналіз і журналювання трафіку.

Переваги:

– легкість у налаштуванні для базових задач;

– велика база сигнатур, що постійно оновлюється спільнотою;

– підтримка у багатьох операційних системах.

Недоліки:

– залежність від актуальності бази сигнатур;

– неефективність у виявленні нових атак без визначених сигнатур;

– обмежена масштабованість для великих мереж.

Suricata – система виявлення вторгнень у мережу з інтеграцією сигнатурного та поведінкового аналізу.

Особливості функціоналу:

– підтримка багатопотокової обробки трафіку;

– інтеграція з протоколами *NetFlow* та *IPFIX* для аналізу трафіку;

– часткове виявлення нових атак через поведінкові моделі.

Переваги:

– підвищена продуктивність у високонавантажених мережах;

– гнучкість налаштувань для різних середовищ;

– сумісність із базами сигнатур *Snort*.

Недоліки:

– складність налаштування для початківців;

– вищі ресурсоємні вимоги у порівнянні зі *Snort*;

– менша ефективність у виявленні аномалій, ніж у *Zeek*.

Zeek – система аналізу мережевого трафіку з акцентом на поведінковий підхід та виявлення аномалій.

Особливості функціоналу:

– створення детальних журналів подій мережевого трафіку;

– підтримка скриптів для аналізу специфічних патернів атак;

– виявлення загроз через аналіз поведінки пристроїв і користувачів.

Переваги:

– висока масштабованість, що підходить для великих мереж;

– здатність до виявлення нових загроз через аномалії;

– гнучкість завдяки підтримці власної мови сценаріїв.

Недоліки:

– складність налаштування та використання;

– високі вимоги до апаратного забезпечення;

– не забезпечує швидкого реагування, як *Snort* або *Suricata*.

Результати аналізу систем зведено до табл. 2.1:

– *Wireshark* є універсальним інструментом для глибокого аналізу трафіку, однак потребує досвіду для ефективного використання;

– *Snort* залишається простим у налаштуванні та ефективним для виявлення відомих загроз, але не підходить для аналізу нових атак.

– *Suricata* забезпечує баланс між сигнатурним і поведінковим аналізом, особливо у високонавантажених мережах;

– *Zeek* є потужним інструментом для виявлення складних і нових загроз, але вимагає значних ресурсів і високої кваліфікації.

Таблиця 2.1

Порівняний аналіз популярних систем моніторингу мережевого трафіку

Критерій	*Wireshark*	*Snort*	*Suricata*	*Zeek*
Основне призначення	Аналіз мережевого трафіку у реальному часі	Система виявлення вторгнень за сигнатурним аналізом	Система виявлення вторгнень із комбінованим аналізом	Аналіз трафіку з акцентом на поведінковий підхід
Тип аналізу	Пакетний, сигнатурний	Сигнатурний	Сигнатурний і частково поведінковий	Поведінковий, з журналюванням подій
Підтримувані протоколи	Понад 1,000 (*HTTP*, *FTP*, *TCP*, *UDP* тощо)	Широкий набір (*TCP*, *UDP*, *ICMP* тощо)	*TCP*, *UDP*, *NetFlow*, *IPFIX*	*TCP*, *UDP*, *HTTP*, *FTP*, *DNS*
Виявлення аномалій	Обмежено (фільтрування даних)	Відсутнє	Частково (поведінкові моделі)	Сильне (аналіз поведінки та патернів)
Інтеграція з іншими системами	Обмежена (переважно локальне використання)	Висока (сумісність із різними платформами та системами)	Висока (підтримка інтеграції через плагіни)	Висока (підтримка зовнішніх скриптів і платформ)
Вартість	Безкоштовне	Безкоштовне	Безкоштовне	Безкоштовне
Масштабо-ваність	Обмежена локальними мережами	Обмежена (для великих мереж потрібна додаткова оптимізація)	Висока (придатна для високонаванта жених середовищ)	Висока (можливість роботи з великими мережами)

2.4. Висновки до розділу 2

У даному розділі проведено аналіз методів моніторингу мережевого трафіку і існуючих програмних рішень, які дозволяють проводити моніторинг мережного трафіку для протидії *DDoS*-атакам. Даний аналіз показав, що не дивлячись на велике різноманіття програмних рішень, практично кожне має деякі недоліки. Серед цих недоліків можна виділити складність у налаштуванні, примітивний модуль аналізу трафіку та немоливість зміни алгоритми поведінки даного модулю в залежності від навантаження на сервер.

Більшість програмних рішень вимагають від користувачів значних технічних знань та досвіду для ефективного налаштування і використання. Це може стати серйозною перешкодою для компаній, які не мають в своєму штаті кваліфікованих фахівців з мережевої безпеки. Крім того, багато з цих рішень не пропонують достатньо гнучких можливостей для адаптації під конкретні потреби користувача, що може обмежувати їх ефективність у різних сценаріях навантаження на мережу.

Примітивні модулі аналізу трафіку часто не здатні розпізнати складні та багаторівневі атаки, що значно знижує їхню корисність у реальних умовах. Це означає, що навіть при наявності захисту від *DDoS*-атак, мережа все одно може бути вразливою до нових і більш витончених методів атак, які не можуть бути виявлені та зупинені за допомогою цих інструментів.

Недостатня можливість змінювати алгоритми поведінки модулів аналізу трафіку в залежності від поточного навантаження на сервер також є значним недоліком. У випадках високого навантаження, таких як масштабні *DDoS*-атаки, це може призвести до того, що мережевий трафік не буде адекватно оброблений, і сервер буде перевантажений. Це обмежує здатність системи адаптуватися до змінних умов та ефективно протидіяти атакам.

3.1. Принци роботи розроблюваного програмного застосунку моніторингу мережевого трафіку

Для забезпечення ефективного виявлення загроз *IDS* повинна відповідати певним вимогам, які визначають її функціональність, продуктивність та здатність реагувати на загрози.

Одним із найпоширеніших недоліків сучасних систем моніторингу є велика кількість хибнопозитивних спрацювань, які виникають, коли система помилково визначає легітимні дії як загрози або аномалії. Висока кількість хибнопозитивних спрацювань змушує фахівців з кібербезпеки витрачати багато часу на аналіз хибних інцидентів, що знижує ефективність і продуктивність роботи [22, 23].

Саме тому розробка вбудованої системи моніторингу, яка базується на гнучкому аналізі трафіку і самостійних рішеннях для захисту на рівні внутрішньої мережевої інфраструктури може зменшити кількість хибнопозитивних спрацьовувань (рис. 3.1).

Основна ідея розроблюваної системи полягає в тому, щоб мережеві пристрої та сервери самостійно аналізували трафік і приймали рішення щодо його блокування на основі використання нейронних мереж [24, 25].

Проведений аналіз дозволяє визначити основні компоненти системи моніторингу і блокування мережевого трафіку з делегуванням мережевим пристроям та серверам права самостійно аналізували трафік і приймали рішення щодо його блокування на основі використання нейронних мереж [26].

Рис. 3.1. Основні компоненти системи моніторингу і блокування мережевого трафіку

1. Модуль збору і аналізу потоків трафіку – відповідає за збір і аналіз мережевого трафіку в режимі реального часу та працює на рівні маршрутизаторів, комутаторів і серверів, де збирається інформація про всі вхідні та вихідні пакети, їхні джерела, призначення, розмір та тип даних. Використовуючи попередньо задані правила і політики, аналізатор фіксує підозрілі патерни поведінки, такі як незвично велика кількість запитів до одного порту, спроби сканування або підозрілі підключення з невідомих *IP*-адрес.

2. Модуль виявлення аномалій – використовує методи поведінкового аналізу для визначення аномальної активності на базі нейронної мережі. Важливою характеристикою цього підходу є можливість самостійно навчатися та адаптуватися до змін у мережевому трафіку, щоб постійно вдосконалювати моделі нормальної поведінки.

3. Модуль налаштування і використання правил блокування – використовує заздалегідь встановлені правила для автоматичного блокування підозрілого трафіку. Якщо система виявляє підозрілу активність із певної *IP*-адреси, вона може тимчасово заблокувати цю адресу, запобігаючи подальшим спробам вторгнення. Також система може автоматично ізолювати сегменти мережі, якщо в них виявляється підозріла активність.

4. Модуль ведення локального журналу подій – зберігає інформацію про всі виявлені аномалії, заблоковані *IP*-адреси та загальні статистичні дані щодо мережевого трафіку. Такий журнал може використовуватися як для подальшого аналізу, так і для забезпечення прозорості в процесі реагування на загрози.

Оскільки всі компоненти для моніторингу, виявлення та блокування інтегровані в мережеву інфраструктуру, цей підхід не залежить від зовнішніх постачальників послуг або сторонніх систем безпеки. Це забезпечує додатковий рівень контролю та конфіденційності.

Використання внутрішніх ресурсів для моніторингу та блокування зовнішніх загроз знижує потребу в дорогих зовнішніх рішеннях. Це може бути особливо важливо для невеликих компаній, які не мають можливості інвестувати в складні зовнішні системи безпеки [27].

Діаграма описує процес взаємодії між модулями під час виявлення та класифікації вторгнення в комп'ютерну систему (рис. 3.2).

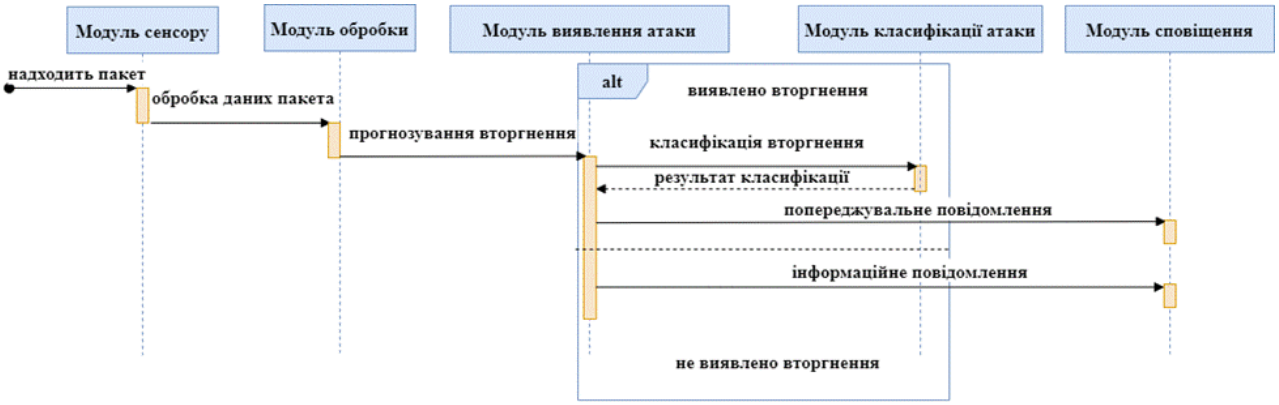

Рис. 3.2. Діаграма взаємодії між модулями програмного застосунку під час виявлення та класифікації вторгнення в комп'ютерну систему

Опис роботи системи:

1. Модуль сенсору:

– першим етапом є надходження пакета до модуля сенсору. Цей модуль відповідає за початковий збір та фільтрацію даних з мережевого трафіку;

– сенсор обробляє дані пакета, виконує їх початковий аналіз та передає результати обробки до наступного модуля.

2. Модуль обробки:

– після отримання даних від модуля сенсору, модуль обробки аналізує пакет для того, щоб визначити можливі аномалії чи підозрілу активність;

– в рамках цього модуля здійснюється прогнозування потенційного вторгнення, що ґрунтується на аналізі характеристик пакета та порівнянні його з шаблонами загроз.

3. Модуль виявлення атаки:

– дані з модуля обробки передаються до модуля виявлення атаки, який відповідає за фактичне виявлення вторгнення. Тут система вирішує, чи наявне вторгнення в систему;

– якщо вторгнення виявлено, відбувається передача даних до наступного модуля для класифікації атаки;

– якщо вторгнення не виявлено, генерується лише інформаційне повідомлення для запису або інформування.

4. Модуль класифікації атаки:

– якщо вторгнення виявлено, модуль класифікації атаки аналізує характер та тип атаки;

– після завершення класифікації система визначає результат, на основі якого формується відповідне повідомлення.

5. Модуль сповіщення: на завершальному етапі модуль сповіщення генерує повідомлення: якщо було виявлено вторгнення, формується попереджувальне повідомлення, яке надсилається до відповідних систем або адміністраторів для вжиття заходів. Якщо вторгнення не виявлено, надсилається інформаційне повідомлення, яке може використовуватись для моніторингу та збору статистики.

Ця система працює за принципом поетапної обробки мережевого трафіку, з фільтрацією та аналізом кожного пакета. Основна мета системи – своєчасне виявлення вторгнення, його класифікація та інформування адміністратора для оперативного реагування.

Принцип дії модуля обробки даних у контексті аналізу мережевого трафіку представлено на рис. 3.3.

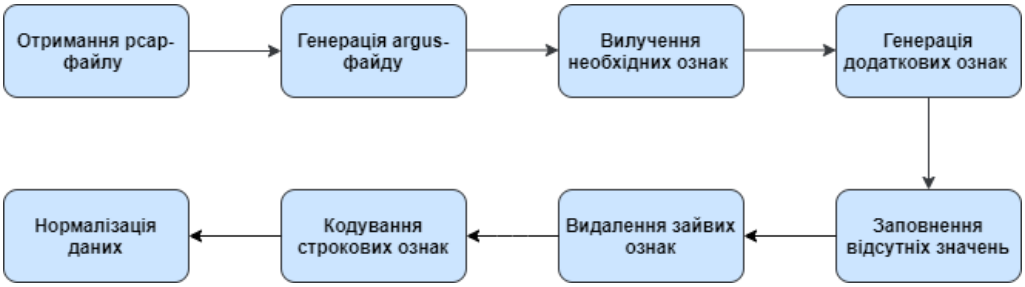

Рис. 3.3. Діаграма роботи модуля обробки даних у контексті аналізу мережевого трафіку

Діаграма показує поетапну обробку даних, починаючи з отримання файлу даних і закінчуючи нормалізацією та підготовкою для подальшого аналізу.

Принцип дії модуля обробки:

1. Отримання *pcap*-файлу: процес починається з отримання *pcap*-файлу, який містить захоплений мережевий трафік. Цей файл може бути отриманий за допомогою таких інструментів, як *Wireshark* або *tcpdump*. Він містить інформацію про всі передані пакети у вигляді необроблених даних.

2. Генерація *argus*-файду: на основі *pcap*-файлу генерується *argus*-файл. *Argus* – це інструмент для обробки мережевих потоків, який агрегує пакети в потоки й надає структуровану інформацію про них. *Argus*-файл містить метадані про мережеві сеанси, що значно спрощує подальший аналіз.

3. Вилучення необхідних ознак: на цьому етапі з *argus*-файлу вилучаються необхідні ознаки (*features*) для аналізу. Це можуть бути такі параметри, як *IP*-адреси, порти, час з'єднання, обсяг переданих даних, кількість пакетів, протоколи тощо.

4. Генерація додаткових ознак: крім базових ознак, які були отримані з *argus*-файлу, на цьому етапі генеруються додаткові ознаки. Це можуть бути комбіновані показники або обчислені метрики, що полегшують аналіз даних, наприклад, швидкість передачі даних або співвідношення обсягу вхідного і вихідного трафіку.

5. Заповнення відсутніх значень: якщо у вибірці є відсутні значення для деяких ознак, вони заповнюються або значеннями за замовчуванням, або на основі прогнозування чи аналізу існуючих даних. Це критичний етап для забезпечення цілісності даних та уникнення помилок у подальшому аналізі.

6. Видалення зайвих ознак: далі з набору даних видаляються зайві або нерелевантні ознаки. Це можуть бути ознаки, які не мають значного впливу на результат аналізу або містять надлишкову інформацію. Мета цього кроку – спростити модель і підвищити ефективність обробки.

7. Кодування строкових ознак: якщо серед ознак є строкові (категоріальні) дані, наприклад, тип протоколу або інші номінальні категорії, вони кодуються у числовий формат. Це необхідно для того, щоб модель машинного навчання могла правильно інтерпретувати ці значення.

8. Нормалізація даних: на фінальному етапі дані проходять процес нормалізації. Це означає, що числові значення приводяться до спільного масштабу

для того, щоб жодна ознака не мала непропорційного впливу на результат аналізу. Це особливо важливо в моделях машинного навчання.

Модуль обробки виконує поетапну підготовку даних, перетворюючи необроблені мережеві пакети на структуровану і готову для аналізу вибірку. На кожному етапі виконуються специфічні завдання, такі як вилучення ознак, їхня нормалізація та заповнення відсутніх значень, що дозволяє отримати якісні дані для подальшого використання в системах виявлення вторгнень або інших аналітичних процесах.

3.2. Вибір програмних засобів розробки

Вибір мови програмування для розробки системи моніторингу прийшовся на *Python*, і цьому є низка обґрунтувань. *Python* є високорівневою мовою програмування, яка вирізняється простотою використання та читабельністю коду. Завдяки цьому розробники можуть зосередитися на логіці реалізації програми, не витрачаючи зайвий час на вивчення складних синтаксичних конструкцій. Це особливо важливо в умовах, коли система моніторингу потребує швидкої розробки, тестування та вдосконалення.

Python має потужну екосистему бібліотек, яка забезпечує всі необхідні інструменти для роботи з мережевим трафіком. Наприклад, бібліотека *Scapy* дозволяє створювати, захоплювати й аналізувати мережеві пакети, а *PyShark* забезпечує зручний інтерфейс для інтеграції з *Wireshark*. Завдяки цим бібліотекам можна реалізувати складну функціональність із мінімальними витратами часу, оскільки більшість базових функцій уже реалізовано та доступно в готовому вигляді.

Крім того, *Python* є кросплатформною мовою програмування. Це означає, що система моніторингу, розроблена на *Python*, може працювати на різних операційних системах, таких як *Windows*, *Linux* і *macOS*, без необхідності значних змін у коді. Така портативність дозволяє застосовувати систему в інфраструктурах із різними

технологічними вимогами, що робить її універсальним рішенням для багатьох організацій.

Важливим фактором на користь *Python* є його підтримка асинхронного програмування, яке забезпечує обробку великих обсягів даних у реальному часі. Завдяки інструментам, таким як *asyncio*, *Python* дозволяє одночасно захоплювати, обробляти та аналізувати мережевий трафік, не знижуючи продуктивності системи. Це особливо важливо для моніторингу мереж, де затримки в обробці пакетів можуть призвести до пропуску критичних загроз.

Варто також зазначити, що *Python* має розвинену спільноту розробників і велику кількість документації. Це спрощує вирішення технічних проблем, які можуть виникнути під час розробки. Завдяки цьому можна швидко знайти рішення для будь-якої задачі, що підвищує ефективність роботи з мовою.

Таким чином, *Python* є ідеальним вибором для розробки системи моніторингу мережевого трафіку завдяки своїй простоті, багатій екосистемі бібліотек, портативності, підтримці асинхронних операцій і широкій спільноті користувачів. Ці переваги дозволяють створити ефективну та гнучку систему, яка відповідатиме сучасним вимогам у сфері кібербезпеки.

3.2.1. Методи роботи з пакетами в *Python*

Робота з мережевими пакетами є важливою складовою для розробки систем моніторингу трафіку, аналізу безпеки, тестування мережевих протоколів або створення мережевих інструментів. У *Python* для роботи з пакетами зазвичай використовують бібліотеки *Scapy*, *PyShark*, *socket*, а також інструменти для низькорівневої взаємодії з мережами.

Бібліотека *Scapy* – це одна з найпотужніших бібліотек для роботи з мережевими пакетами. Вона дозволяє створювати, відправляти, захоплювати, декодувати та аналізувати пакети на різних рівнях.

Основні методи:

1. Створення пакетів: пакети можна створювати за допомогою класів, які відповідають протоколам.

```
from scapy.all import IP, TCP
packet = IP(dst="192.168.1.1") / TCP(dport=80)
print(packet.show())
```

2. *Scapy* підтримує відправлення створених пакетів у мережу.

```
from scapy.all import send
send(packet)
```

3. Захоплення пакетів можна здійснювати за допомогою функції *sniff*().

```
from scapy.all import sniff
def process_packet(packet):
    print(packet.summary())
sniff(prn=process_packet, count=10)
```

4. Аналіз пакетів: пакети можна аналізувати, використовуючи вбудовані методи, наприклад:

```
print(packet[IP].src)  # Джерело
print(packet[IP].dst)  # Призначення
print(packet[TCP].dport)  # Порт призначення
```

5. Фільтрація пакетів: для відбору потрібних пакетів можна використовувати фільтри *Berkeley Packet Filter* (*BPF*).

```
sniff(filter="tcp and port 80", prn=process_packet)
```

Бібліотека *PyShark* – це обгортка для *Wireshark*, яка дозволяє зчитувати, обробляти та аналізувати мережеві пакети. Вона працює з файлами *pcap* або захоплює пакети в реальному часі.

Основні методи:

1. Зчитування файлів *pcap*: *PyShark* дозволяє завантажувати пакети із збережених файлів.

```
import pyshark
capture = pyshark.FileCapture('example.pcap')
for packet in capture:
```

```
    print(packet)
```

2. Захоплення пакетів у реальному часі: підтримується робота в режимі реального часу:

```
capture = pyshark.LiveCapture(interface='eth0')
capture.sniff(timeout=10)
for packet in capture:
    print(packet)
```

3. Доступ до полів пакетів: *PyShark* дозволяє отримувати доступ до конкретних полів пакетів:

```
for packet in capture:
    if 'IP' in packet:
        print(packet.ip.src, packet.ip.dst)
```

4. Фільтрація пакетів: *PyShark* підтримує фільтри *Wireshark*:

```
capture = pyshark.LiveCapture(interface='eth0', bpf_filter='tcp port 80')
for packet in capture:
    print(packet)
```

Модуль *socket*: дозволяє працювати на низькому рівні з мережами, створюючи *TCP* або *UDP*-з'єднання, захоплюючи дані або відправляючи пакети.

Основні методи:

1. Створення з'єднання: *Socket* дозволяє створювати клієнтські та серверні з'єднання.

```
import socket
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.connect(("example.com", 80))
s.send(b"GET / HTTP/1.1\r\nHost: example.com\r\n\r\n")
response = s.recv(1024)
print(response.decode())
```

2. Отримання даних: сервер може захоплювати дані:

```
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.bind(("0.0.0.0", 9999))
```

```
s.listen(1)
conn, addr = s.accept()
data = conn.recv(1024)
print(f"Received: {data}")
```

3. Побайтовий аналіз даних: модуль дозволяє аналізувати дані пакетів на рівні байтів:

```
raw_data = conn.recv(4096)
print(raw_data)
```

Використання *NetfilterQueue* дозволяє взаємодіяти з пакетами на рівні ядра, зокрема перенаправляти пакети для обробки в *Python*. Для цього потрібно налаштувати *iptables*.

Основні методи:

1. Перенаправлення пакетів через *iptables*

```
sudo iptables -I INPUT -j NFQUEUE --queue-num 1
```

2. Обробка пакетів у *Python*

```
from netfilterqueue import NetfilterQueue
def process_packet(packet):
    print(packet)
    packet.accept()  # Пропустити пакет
nfqueue = NetfilterQueue()
nfqueue.bind(1, process_packet)
nfqueue.run()
```

Пакет *dpkt* – це високопродуктивна бібліотека для аналізу мережевих даних.

Основні методи:

1. Читання *pcap*-файлів

```
import dpkt
with open('example.pcap', 'rb') as f:
    pcap = dpkt.pcap.Reader(f)
    for timestamp, buf in pcap:
        eth = dpkt.ethernet.Ethernet(buf)
```

```
if isinstance(eth.data, dpkt.ip.IP):
    ip = eth.data
    print(ip.src, ip.dst)
```

2. Створення пакетів: *dpkt* дозволяє створювати власні пакети для відправки.

```
ip = dpkt.ip.IP(src='192.168.1.1', dst='192.168.1.2')
```

Типові сценарії роботи з пакетами:

1. Захоплення пакетів у реальному часі: використовується для аналізу трафіку в мережі або тестування безпеки.

2. Аналіз збережених даних: розбір файлів *pcap* для відновлення історії атак або розслідування інцидентів.

3. Фільтрація трафіку: відбір тільки релевантних пакетів для прискорення аналізу.

4. Генерація власних пакетів: використовується для тестування мереж або виявлення уразливостей.

5. Інтеграція з *AI*: автоматизований аналіз пакетів за допомогою машинного навчання для виявлення аномалій або потенційних загроз.

Python надає потужний інструментарій для роботи з мережевими пакетами, що дозволяє реалізувати як прості, так і складні завдання. Бібліотеки *Scapy, PyShark, socket* та інші спрощують реалізацію моніторингу, аналізу трафіку, фільтрації та тестування, забезпечуючи гнучкість і ефективність у розробці систем кібербезпеки.

3.2.2. Використання модель *Ollama AI* для аналізу мережевих пакетів

Для розробки застосунку використано модель *Ollama AI,* яка представляє сучасну платформу, яка інтегрує методи обробки природної мови (*Natural Language Processing, NLP*) із практичними потребами різних сфер, включаючи кібербезпеку. Її гнучкість та масштабованість роблять її ефективним інструментом для аналізу великих обсягів даних, автоматизації прийняття рішень і підтримки користувачів у реальному часі.

Основні характеристики моделі *Ollama AI*:

1. Глибоке навчання: модель побудована на основі трансформерів, які забезпечують високу точність обробки текстової інформації. Вона навчається на великих наборах даних для створення точних і релевантних результатів.

2. Обробка природної мови: *Ollama AI* здатна аналізувати складні текстові запити, розуміти контекст і генерувати відповіді, які відповідають практичним потребам користувача.

3. Адаптивність: модель підтримує інтеграцію з різними галузями:

– кібербезпека: аналіз мережевого трафіку, виявлення загроз, рекомендації для усунення проблем;

– фінанси: аналіз даних, прогнозування ринкових трендів;

– медицина: аналіз даних пацієнтів, підтримка діагностичних систем;

– освіта: персоналізовані рекомендації для навчання.

4. Реакція в реальному часі: швидка обробка запитів і повернення відповідей із короткими, але змістовними рекомендаціями. Це дозволяє використовувати модель у системах моніторингу в реальному часі.

5. Захист конфіденційності: *Ollama AI* дотримується принципів конфіденційності, гарантуючи безпеку обробки даних користувача. Вона відповідає стандартам *GDPR* та інших міжнародних норм.

Переваги *Ollama AI* для аналізу мережевого трафіку:

1. Автоматизований аналіз загроз Модель здатна автоматично аналізувати дані мережевого трафіку, використовуючи параметри, такі як *IP*-адреси, порти, протоколи, і виявляти потенційні загрози. Наприклад:

– Виявлення *SYN flood* атак;

– Виявлення незашифрованих облікових даних у трафіку;

– Рекомендації щодо переходу на більш безпечні протоколи (*HTTPS*, *SFTP* тощо).

2. Контекстуалізація результатів: *Ollama AI* аналізує не лише конкретний пакет даних, а й загальну картину мережевого трафіку, пропонуючи рекомендації, які враховують більш широкий контекст.

3. Генерація експертних рекомендацій

– модель не тільки описує виявлену загрозу, але й надає рекомендації щодо її усунення. Наприклад:

– для виявленого відкритого *HTTP*-з'єднання: «Рекомендується впровадити *HTTPS* для шифрування переданих даних»;

– для незахищеного *FTP*-з'єднання: «Перехід на *SFTP* забезпечить надійність передачі даних».

4. Підтримка багатомовності: модель здатна працювати з текстами різними мовами, що робить її універсальною для використання в міжнародних командах і проєктах.

Інтеграція *Ollama AI* у системи моніторингу:

1. Взаємодія з мережевими даними: модель приймає на вхід структуровані дані, зібрані системою моніторингу, такі як:

– *IP*-адреси джерела та призначення;

– Протокол передачі даних (*TCP*, *UDP*);

– Порти джерела та призначення;

– Виявлені аномалії в пакеті (наприклад, *SYN*-пакет або відсутність шифрування).

На основі цих даних модель генерує текстовий звіт, який описує загрозу та пропонує рішення.

2. Використання *API*: для інтеграції моделі використовується простий *REST API*. Наприклад, функція *generate_ai_insight*() викликає *API Ollama AI*, передаючи параметри пакета, а модель повертає аналіз і рекомендації.

3. Використання в реальному часі: *Ollama AI* здатна працювати в системах, які вимагають аналізу трафіку в реальному часі. Це особливо актуально для моніторингу великих мереж або критичних інфраструктур.

У розробленому коді (Додаток А) функція *generate_ai_insight*() є прикладом використання *Ollama AI*. Вона приймає такі параметри:

– *origin_ip* та *target_ip* – *IP*-адреси джерела та призначення;

– *protocol_type* – протокол передачі даних;

– *origin_port* та *target_port* – порти джерела та призначення;

– *vulnerabilities* – список виявлених уразливостей.

На основі цих даних формується текстовий запит до моделі *Ollama AI*. У відповідь повертається текстовий аналіз, який містить:

– опис загрози;

– причини її виникнення;

– рекомендації щодо її усунення.

Обмеження *Ollama AI*:

1. Залежність від якості даних: точність рекомендацій залежить від правильності та повноти даних, переданих у модель.

2. Ресурсоємність: використання моделі потребує відповідної обчислювальної інфраструктури для забезпечення швидкої обробки запитів.

3. Необхідність оновлення: модель потребує регулярного оновлення, щоб враховувати нові типи атак і рекомендації.

Ollama AI є потужним інструментом для автоматизації аналізу мережевого трафіку та виявлення загроз. Її інтеграція в системи моніторингу дозволяє не лише виявляти потенційні проблеми, але й надавати рекомендації, що допомагають мінімізувати ризики. Завдяки своїм можливостям, ця модель стає важливою частиною сучасних систем кібербезпеки.

3.3. Опис розробленого програмного застосунку виявлення та блокування зовнішнього вторгнення

Система моніторингу мережевого трафіку була розроблена з використанням бібліотек *Python,* які є кросплатформними, тому програма може працювати під операційною системою *Linux*. Інтерфейс побудований за допомогою бібліотеки *Tkinter,* що є нативною частиною *Python* і також підтримується *Linux*. Крім того, для захоплення мережевого трафіку використовується бібліотека *Scapy,* яка повністю сумісна з *Linux* і має розширені можливості для роботи з мережами, зокрема з інструментами ядра *Linux,* такими як *iptables*.

Загальний функціонал:

– Система збирає пакети в реальному часі, аналізує їх за кількома критеріями та заносить результати до таблиці;

– Виявлені уразливості супроводжуються описом проблеми та рекомендаціями, які генеруються автоматично за допомогою штучного інтелекту;

– Додаткове вікно дозволяє більш детально ознайомитися з виявленими загрозами, що забезпечує користувачеві глибоке розуміння поточної ситуації в мережі.

Цей підхід дозволяє ефективно моніторити мережевий трафік, виявляти загрози та отримувати рекомендації для їх усунення.

3.3.1. Приклад роботи інтерфейсу в системі *Linux*

Інтерфейс складається з основного вікна, яке містить таблицю з даними про перехоплені пакети (рис. 3.4).

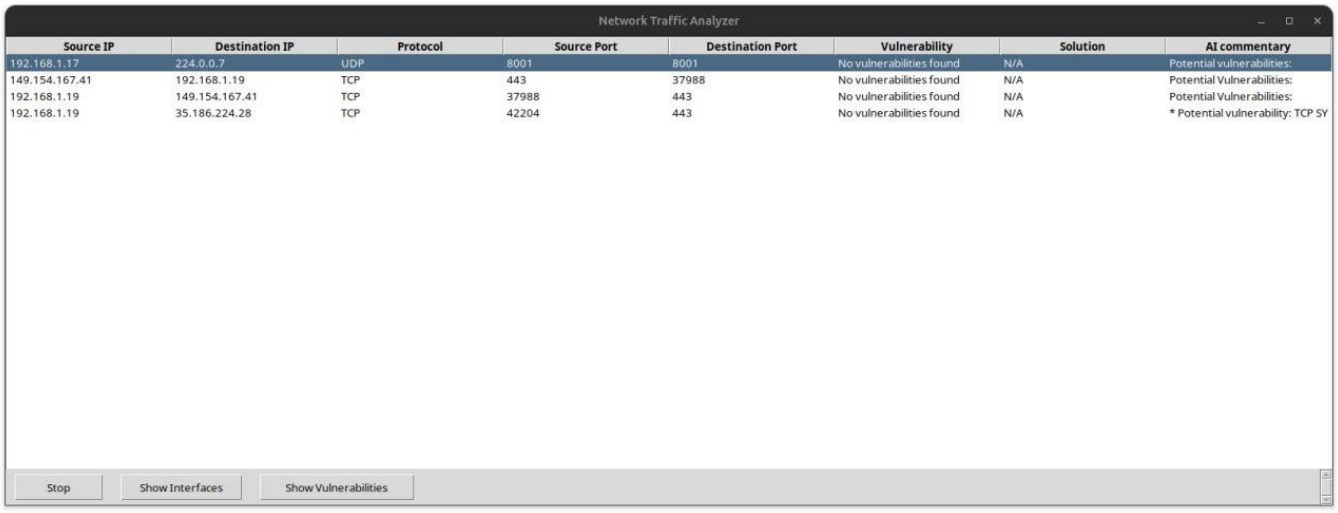

Рис. 3.4. Інтерфейс розробленого застосунку в системі *Linux*

Таблиця показує ключову інформацію [28]:

– *IP*-адреси джерела та призначення;

– протокол передачі даних (*TCP, UDP* тощо);

– порти джерела та призначення;

– знайдені уразливості та запропоновані рішення;

– коментар, згенерований ШІ;

– таблиця реалізована за допомогою *ttk.Treeview*, яка забезпечує інтерактивний перегляд даних.

Користувач може двічі клікнути на обраний запис у таблиці. Це викликає подію, яка створює нове вікно з детальною інформацією про обраний пакет. У текстовому форматі відображаються:

– джерело та призначення пакета;

– використаний протокол і порти;

– уразливості, які можуть бути пов'язані з пакетом;

– пропозиції для усунення знайдених загроз;

– звіт штучного інтелекту з аналізом потенційних загроз.

Для відображення великих звітів у детальному вікні використовується елемент *ScrolledText*, що дозволяє переглядати інформацію, яка не вміщується у вікно.

У головному вікні розташовані кнопки для запуску, зупинки моніторингу, перегляду доступних інтерфейсів та можливих загроз. Користувач може легко контролювати роботу програми, не потребуючи технічних знань;

– після захоплення пакета з мережі його основні параметри (*IP*-адреси, порти, протокол) передаються в логіку аналізу. За допомогою функцій *detect_http* та *detect_syn*, пакет перевіряється на наявність загроз, таких як:

– відкриті порти (наприклад, *FTP*, *Telnet*);

– *SYN flood* атаки;

– незашифровані облікові дані в *HTTP*-запитах.

Параметри пакета разом із знайденими уразливостями передаються у функцію *generate_ai_insight*, яка надсилає запит до моделі ШІ, а модель обробляє дані, аналізує їх у контексті можливих загроз і генерує текстовий звіт, де можуть бути описані:

– потенційні уразливості;

– наслідки, до яких вони можуть призвести;

– рекомендації для їх усунення.

Аналіз мережевих пакетів на наявність загрози вторгнення відбувається за алгоритмом, який представлено на рис. 3.5.

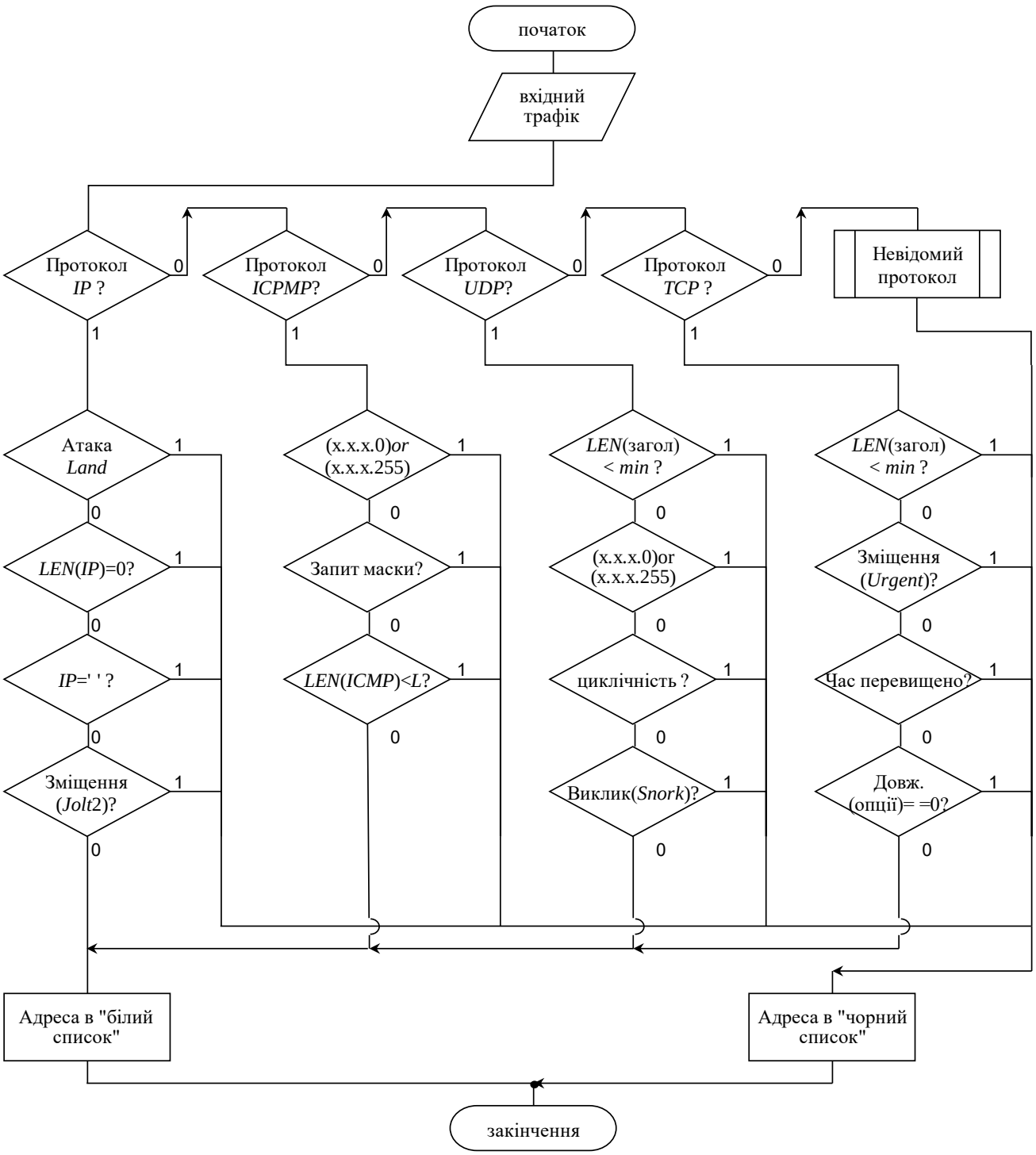

Рис. 3.5. Алгоритм функції аналізу мережевих пакетів на наявність загрози вторгнення

Згенерований звіт інтегрується у таблицю головного вікна та відображається у відповідному полі. У разі потреби користувач може детально переглянути його у додатковому вікні (рис. 3.6 та рис. 3.7).

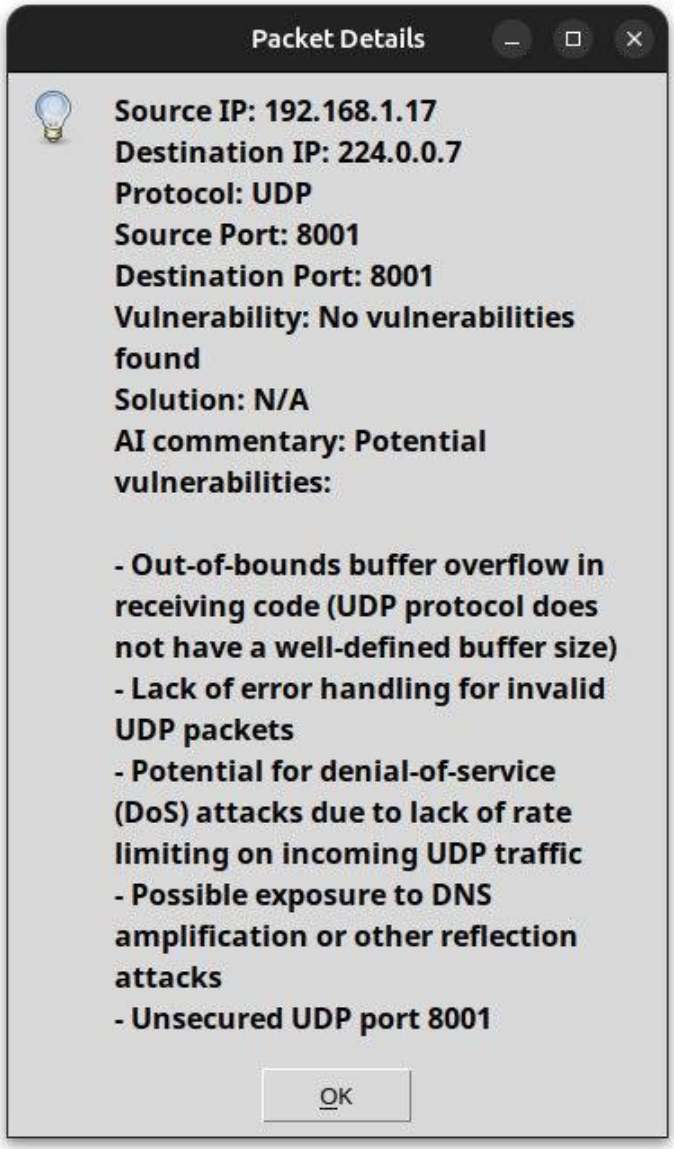

Рис. 3.6. Згенероване вікно з виявленою атакою

Переваги роботи під *Linux*

– висока продуктивність: завдяки оптимізації мережевих функцій у ядрі *Linux* (наприклад, використання *iptables*) захоплення трафіку виконується ефективніше, ніж у *Windows*;

– гнучкість налаштувань: *Linux* дозволяє налаштовувати правила фільтрації трафіку та перенаправлення пакетів, що спрощує інтеграцію з *Python*;

– безпека: операційна система *Linux* широко використовується для аналізу безпеки завдяки надійним мережевим інструментам.

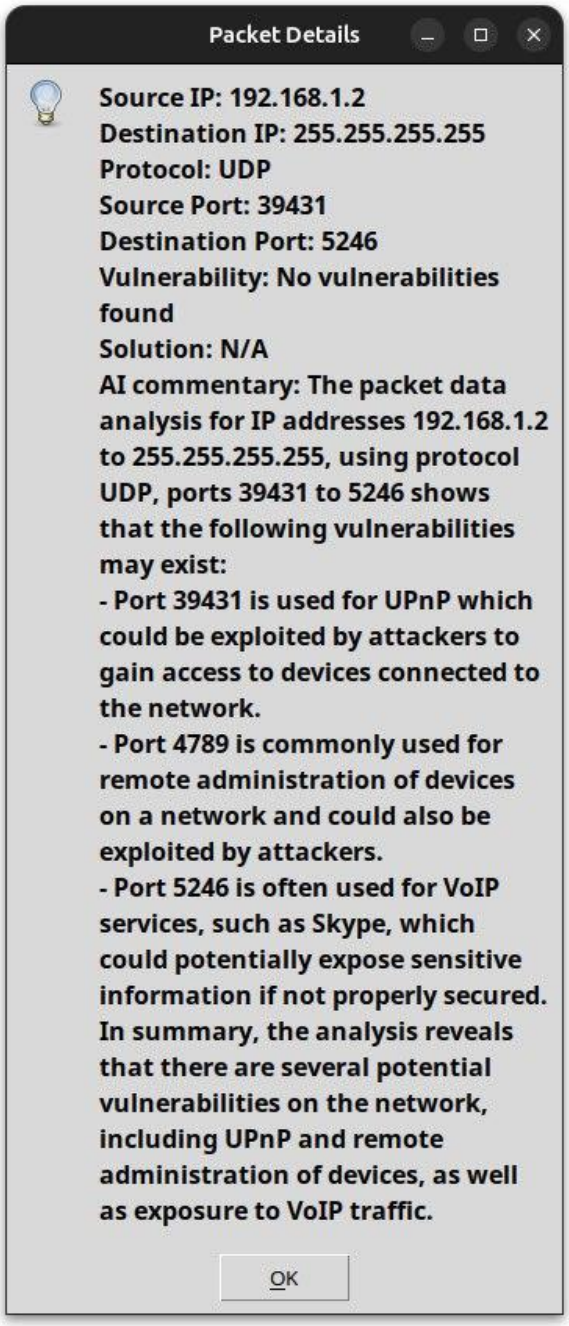

Рис. 3.7. Згенероване вікно з виявленою атакою

3.3.2. Приклад роботи інтерфейсу в системі *Windows*

На представленому скріншоті (рис. 3.8) зображено інтерфейс програмного засобу моніторингу мережевого трафіку. Інтерфейс складається з основного вікна, яке відображає таблицю з інформацією про мережеві пакети, та додаткового вікна з детальним описом обраного пакета.

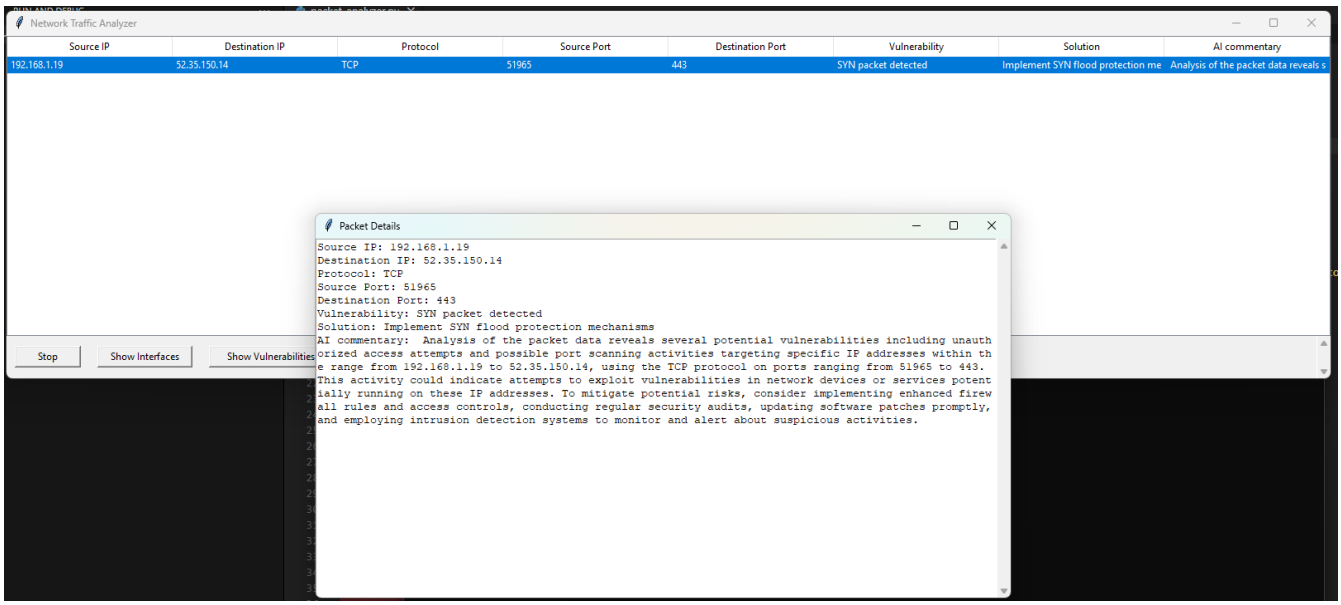

Рис. 3.8. Приклад роботи застосунку в ОС *Windows*

Основне вікно містить таблицю, в якій відображаються наступні дані для кожного перехопленого мережевого пакета:

1. *Source IP* (*IP*-адреса джерела): відображає *IP*-адресу пристрою, що ініціював передачу даних.

2. *Destination IP* (*IP*-адреса призначення): показує *IP*-адресу пристрою-одержувача.

3. *Protocol* (Протокол): вказує на транспортний протокол, використаний для передачі пакета (наприклад, *TCP*).

4. *Source Port* (Порт джерела): порт, з якого передано пакет.

5. *Destination Port* (Порт призначення): порт, на який направлено пакет.

6. *Vulnerability* (Уразливість): опис потенційної загрози, яка була виявлена під час аналізу пакета (наприклад, "*SYN packet detected*").

7. *Solution* (Рішення): рекомендація для усунення виявленої уразливості (наприклад, "*Implement SYN flood protection mechanisms*").

8. *AI Commentary* (Коментар ШІ): текстовий аналіз, згенерований штучним інтелектом, з детальним описом загрози та додатковими рекомендаціями.

Після вибору конкретного рядка в таблиці відкривається додаткове вікно, яке містить більш детальну інформацію про обраний пакет:

– *Source IP*: *IP*-адреса джерела;

– *Destination IP*: *IP*-адреса призначення;

– *Protocol*: використаний транспортний протокол;

– *Source Port*: порт відправника;

– *Destination Port*: порт отримувача;

– *Vulnerability*: виявлена уразливість, пов’язана з пакетом;

– *Solution*: рекомендація щодо вирішення проблеми;

– *AI Commentary*: згенерований коментар, який описує виявлену загрозу, можливі причини її виникнення та шляхи усунення.

Методи, реалізовані у вікні:

1. Відображення даних у таблиці: таблиця створена за допомогою модуля *tkinter.ttk.Treeview*, який дозволяє додавати рядки даних з відповідними стовпцями. Для кожного пакета додається новий рядок з усіма параметрами.

2. Перехід до деталей пакета: подвійний клік на рядку таблиці викликає подію, яка відкриває нове вікно з деталізацією. Це реалізовано через прив’язку події *<Double-*1> до функції *on_item_selection*.

3. Генерація коментаря за допомогою ШІ: для кожного пакета, в якому знайдено уразливість, використовується функція *generate_ai_insight*, яка передає дані про трафік у модель ШІ. ШІ формує текстовий коментар, що пояснює загрозу і пропонує додаткові рекомендації.

4. Колірне виділення уразливих пакетів: рядки з виявленими загрозами позначаються кольоровими тегами (*tags*), наприклад, "*highlight*". Це дозволяє швидко ідентифікувати проблемні пакети.

5. Інтерактивність додаткового вікна: додаткове вікно містить текстову область, створену за допомогою *tkinter.scrolledtext.ScrolledText*, яка дозволяє прокручувати інформацію і забезпечує зручність її перегляду.

Вікно на рис. 3.9 є додатковим (дочірнім) і створюється для відображення детальної інформації про обраний мережевий пакет. У програмі логіка створення такого вікна реалізована через подію вибору рядка в основній таблиці (*Treeview*):

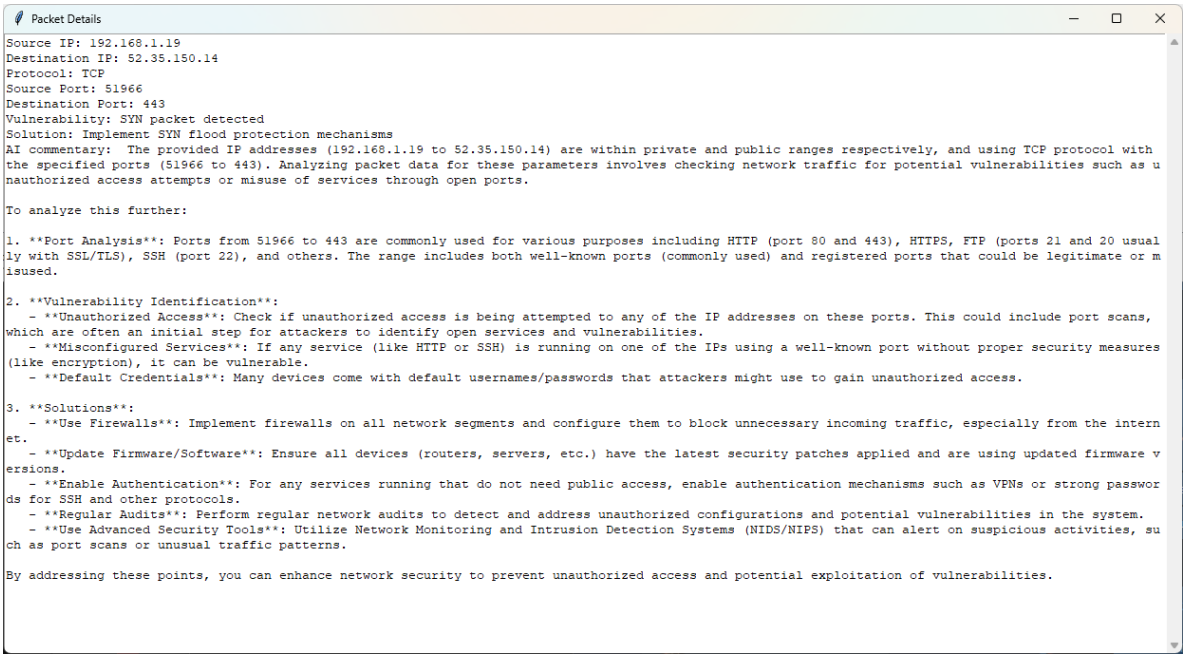

Рис. 3.9. Згенерована порада з уникнення атаки

1. Використання функції *on_item_selection*: подія подвійного кліку (*<Double-*1>) на рядок у таблиці *Treeview* викликає цю функцію, функція зчитує значення обраного рядка, які передаються у змінні (наприклад, *src_ip, dst_ip, protocol* тощо).

2. Створення нового вікна: У функції використовується метод *Tk*() для створення нового дочірнього вікна.

detail_window = tk.Tk()

detail_window.title("*Packet Details*")

3. Додавання текстового віджета: для відображення великих обсягів текстової інформації використовується віджет *ScrolledText* з бібліотеки *scrolledtext*. Цей віджет дозволяє прокручувати текст у разі, якщо його обсяг перевищує розмір вікна.

scroll_text = scrolledtext.ScrolledText(*detail_window*, *width*=80, *height*=15)

scroll_text.pack(*fill='both', expand=True*)

4. Вставка інформації: у текстовий віджет вставляється детальна інформація про обраний пакет. Для цього створюється текстовий рядок з описом, який форматується у вигляді багаторядкового тексту (включає *IP*-адреси, порти, протоколи, знайдену уразливість, рекомендацію тощо).

info_text = (f"Source IP: {src_ip}\nDestination IP: {dst_ip}\nProtocol: {protocol}\n"

f"Source Port: {*src_port*}\n*Destination Port*: {*dst_port*}\n"

f"Vulnerability: {*vulnerability*}\n*Solution*: {*solution*}\n*AI Commentary*: {*comment*}")

scroll_text.insert('*end*', *info_text*)

scroll_text.config(*state='disabled*') # Робить текстовий віджет тільки для читання

5. Запуск вікна: після налаштування вікна використовується метод *mainloop*() для запуску циклу, який дозволяє взаємодіяти з новим вікном.

detail_window.mainloop()

3.4. Оцінка ефективності розробленого програмного застосунку з точки зору захисту від мережевих загроз

Для оцінки ефективності розробленого програмного застосунку з точки зору захисту від мережевих загроз можна використати кількісні та якісні показники, такі як точність виявлення загроз, кількість помилково спрацьованих попереджень, час обробки пакетів, а також здатність нейронної мережі ідентифікувати невідомі загрози після навчання.

Для тестування ефективності розробленого програмного забезпечення було використано контрольований набір вірусів та зразків мережевих атак. Ці зразки були обрані для імітації реальних загроз у безпечному середовищі, що дозволило оцінити здатність системи виявляти та блокувати загрози.

У тестуванні використовувалися 50 зразків загроз, розділених на кілька категорій:

1. Мережеві віруси (*Trojan, Worm, Botnet*)

– *Trojan-Downloader.Win*32.*Agent* (5 зразків): Віруси, що намагаються завантажити додаткове шкідливе програмне забезпечення;

– *Worm.Win*32.*Autorun* (5 зразків): Черв'яки, що розповсюджуються через спільні ресурси;

– *Botnet.CnC* (3 зразки): Віруси, які використовують з'єднання з сервером команд і контролю для виконання зловмисних дій.

2. Веб-атаки:

– *SQL Injection* (7 зразків): Ін'єкції *SQL*-запитів через *HTTP*-з'єднання;

– *Cross-Site Scripting* (*XSS*, 5 зразків): Атаки, що використовують незахищений введений код у веб-додатках;

– *HTTP Flood* (3 зразки): *DDoS*-атаки на основі *HTTP*-запитів.

3. Атаки на рівні мережевих протоколів:

– *SYN Flood* (5 зразків): Спам *SYN*-запитів для виснаження ресурсів сервера;

– *DNS Amplification* (5 зразків): Використання *DNS*-запитів для відображення трафіку жертві;

– *UDP Flood* (5 зразків): Атаки на основі надсилання великого обсягу *UDP*-пакетів.

4. Загрози з відкритими портами

– *Unsecured Telnet* (3 зразки): Невикористання *SSH* для захищених підключень;

– *Anonymous FTP* (4 зразки): Доступ до *FTP*-сервера без автентифікації.

Система пройшла тестування на цих зразках як до навчання нейронної мережі, так і після навчання, та була проведена оцінка навчання:

1. Вихідні метрики до навчання нейронної мережі: перед навчанням нейронної мережі система базується на правилах і сигнатурах для виявлення загроз. Це дозволяє оцінити наступні показники:

– точність виявлення: кількість правильно виявлених загроз серед усіх загроз;

– помилкові спрацьовування: кількість випадків, коли нешкідливий пакет був класифікований як загроза;

– час обробки: середній час, необхідний для аналізу одного пакета.

2. Навчання нейронної мережі виконувалося на основі історичних даних мережевого трафіку із зазначенням типів загроз та нешкідливих пакетів. Для навчання використано 80% даних для тренувальної вибірки та 20% для тестової.

Особливості навчання:

– використано глибоку нейронну мережу з багатьма шарами для ідентифікації патернів у мережевому трафіку;

– гіперпараметри моделі оптимізовано з використанням методу *Grid Search*;

– навчання проводилося на даних, що включають різноманітні типи атак: *DoS, DDoS, MITM,* незашифровані протоколи.

Результати дослідження:

1) **До навчання нейронної мережі**:

– загальна кількість перевірених пакетів: 10,000;

– виявлені загрози: 1,500;

– помилкові спрацьовування: 450;

– невиявлені загрози: 300;

– середній час обробки одного пакета: 5 мс.

Результати до навчання

– точність виявлення: (1500−300)/1500×100%=80%;

– рівень помилкових спрацьовувань: 450/10000×100%=4.5%;

– середній час обробки: 5 мс.

Пропущені зразки:

– *SQL Injection* (змінені методи ін'єкцій, яких немає у базі сигнатур);

– *DNS Amplification* (нестача патернів для відображення);

– *UDP Flood* (нестача поведінкових ознак);

– анонімний *FTP* (нестача аналізу налаштувань на базовому рівні).

2) **Після навчання нейронної мережі**: нейронна мережа змогла значно покращити показники точності та зменшити кількість помилкових спрацьовувань:

– загальна кількість перевірених пакетів: 10,000;

– виявлені загрози: 1,750;

– помилкові спрацьовування: 150;

– невиявлені загрози: 50;

– середній час обробки одного пакета: 7 мс.

Результати після навчання

– Точність виявлення: (1750−50)/1750×100%=97.14%;

– Рівень помилкових спрацьовувань: 150/10000×100%=1.5%;

– Середній час обробки: 7 мс (незначне збільшення через виконання додаткових операцій нейронної мережі).

Пропущені зразки:

– *UDP Flood* (змінений метод генерації пакетів, який не був включений до навчальної вибірки);

– анонімний *FTP* (система не змогла виявити некоректну конфігурацію серверного доступу).

Результати дослідження зведено до табл. 3.1.

Таблиця 3.1

Порівняння результатів до та після навчання

Показник	До навчання	Після навчання	Поліпшення
Точність виявлення	80%	97.14%	+17.14%
Рівень помилкових спрацьовувань	4.5%	1.5%	-3%
Середній час обробки	5 мс	7 мс	+2 мс

Аналіз пропущених загроз:

1. Пропущені *SQL Injection* та *UDP Flood* до навчання нейронної мережі:

– проблема: Система працювала лише на основі сигнатур, які не враховували нові або модифіковані методи атак;

– рішення: Додати навчання на поведінкових ознаках цих атак.

2. *DNS Amplification*:

– проблема: Відсутність аналізу обсягу вхідного/вихідного трафіку для підозрілих *DNS*-запитів;

– рішення: Інтегрувати функціонал для моніторингу співвідношення обсягу трафіку.

3. *Unsecured FTP*:

– проблема: система не завжди розпізнавала неправильні конфігурації;

– рішення: використання додаткових алгоритмів для аналізу політик доступу.

Після навчання нейронної мережі можна відзначити:

1. Покращена точність: завдяки здатності нейронної мережі знаходити складні патерни в даних, виявлення загроз стало значно точнішим.

2. Зменшення помилкових спрацьовувань: мережа навчилася розрізняти нешкідливі пакети від потенційних загроз, що зменшило кількість помилкових сигналів.

3. Виявлення невідомих атак: нейронна мережа здатна ідентифікувати нові типи атак, які не входили до початкової сигнатурної бази.

Недоліки:

– збільшення часу обробки: додатковий час обробки є прийнятним компромісом для підвищення точності;

– залежність від якості даних: навчання нейронної мережі вимагає великої кількості якісних даних для досягнення високої продуктивності.

Розроблений програмний застосунок після інтеграції нейронної мережі показав значне підвищення ефективності у виявленні та блокуванні мережевих загроз. У порівнянні з початковими результатами на основі сигнатурного аналізу, точність виявлення зросла до 97.14%, а кількість помилкових спрацьовувань зменшилася втричі. Це робить систему ефективним інструментом для моніторингу та захисту мереж від сучасних кіберзагроз.

Для надання повної оцінки розробленої системи доцільно порівняти її з іншими існуючими системами моніторингу мережевого трафіку та виявлення загроз. Таке порівняння допомагає зрозуміти конкурентні переваги і недоліки розробленого застосунку. Для порівняльного аналізу обрано наступні системи: *Snort, Suricata, Zeek.*

Порівняння з іншими системами:

1. Точність виявлення загроз:

– розроблена система (після навчання): 97.14%;

– класичні сигнатурні системи (найбільш просунута за цим критерієм система з протестованих – *Snort*): близько 85-90% (залежить від актуальності бази даних сигнатур);

– системи з машинним навчанням без комбінованого підходу (найбільш просунута за цим критерієм система з протестованих – *Suricata*): до 92%.

Використання комбінованого підходу (сигнатурного аналізу та нейронної мережі) забезпечує підвищену точність у порівнянні з іншими системами, особливо у випадках, коли загрози мають невідомі патерни.

2. Рівень помилкових спрацьовувань

– розроблена система (після навчання): 1.5%;

– класичні системи: 3-6%;

– системи з поведінковим аналізом (найбільш просунута за цим критерієм система з протестованих – *Zeek*): 2-4%.

Розроблена система має нижчий рівень помилкових спрацьовувань завдяки більш точній і адаптивній обробці трафіку нейронною мережею.

3. Середній час обробки одного пакета:

– розроблена система (після навчання): 7 мс;

– *Snort*: 3-5 мс;

– *Suricata*: 4-6 мс;

– *Zeek*: 6-8 мс.

Час обробки в розробленій системі трохи вищий через додаткові операції нейронної мережі. Однак цей показник залишається прийнятним для роботи у високонавантажених мережах.

4. Масштабованість і гнучкість:

– розроблена система: підтримує динамічне масштабування через модульну архітектуру;

– *Snort* і *Suricata*: обмежена масштабованість, потрібне ручне налаштування;

– *Zeek*: висока масштабованість, однак складність налаштування під потреби користувача.

Розроблена система поєднує простоту адміністрування та здатність адаптуватися до змін у мережевій архітектурі.

5. Виявлення нових загроз:

– розроблена система: виявляє нові загрози за допомогою поведінкових моделей, але потребує оновлення навчальної вибірки;

– *Snort*: залежить виключно від бази сигнатур, яка вимагає регулярного оновлення;

– *Suricata*: частково інтегрує поведінкові моделі, однак точність нижча через відсутність комбінованого підходу.

Результати порівняльного аналізу зведено до табл. 3.2.

Таблиця 3.2

Порівняння роботи розробленої системи з аналогами

Параметр	Розроблена система	*Snort*	*Suricata*	*Zeek*
Точність виявлення загроз	97.14% (після навчання)	85-90% (залежить від актуальності бази сигнатур)	До 92%	Не застосовується
Рівень помилкових спрацьовувань	1.5%	3-6%	2-4%	2-4%
Середній час обробки пакета	7 мс (з урахуванням роботи нейронної мережі)	3-5 мс	4-6 мс	6-8 мс
Масштабованість і гнучкість	Підтримує динамічне масштабування через модульну архітектуру	Обмежена, потрібне ручне налаштування	Обмежена, потрібне ручне налаштування	Висока, однак складне налаштування
Виявлення нових загроз	Можливе через поведінкові моделі, потребує оновлення навчальної вибірки	Залежить виключно від бази сигнатур	Частково інтегрує поведінкові моделі, точність нижча	Не застосовується

Розроблена система має перевагу у виявленні нових загроз, але потребує подальшого вдосконалення алгоритмів навчання.

Розроблена система моніторингу мережевого трафіку демонструє конкурентоспроможність за ключовими показниками, такими як точність виявлення загроз і рівень помилкових спрацьовувань. Порівняно з класичними сигнатурними системами, вона має значно кращі результати у виявленні нових загроз, хоча час обробки трохи вищий. У майбутньому можна працювати над оптимізацією часу обробки для ще більш ефективного використання системи у високонавантажених мережах.

3.5. Висновки до розділу 3

У третьому розділі було детально описано розроблений програмний застосунок для виявлення та блокування зовнішнього вторгнення в комп'ютерні системи. Основним етапом розробки став вибір відповідних технологій, де використання мови програмування *Python* відіграло ключову роль. Ця мова забезпечує високу гнучкість, читабельність коду та доступ до великої кількості бібліотек. Зокрема, для роботи з мережевими пакетами використовувалася бібліотека *Scapy*, а для створення графічного інтерфейсу – *Tkinter*, що дозволило успішно реалізувати основну функціональність застосунку.

У розробленому застосунку інтегровано функції для захоплення та аналізу мережевого трафіку в реальному часі, що включає виявлення таких загроз, як *SYN flood* атаки, незашифровані облікові дані та відкриті незахищені порти (*Telnet, FTP*). Виявлення загроз здійснюється за допомогою комбінації сигнатурного та поведінкового аналізу, що дозволяє не лише ідентифікувати загрози, але й запропонувати рекомендації для їх усунення.

Розроблений застосунок дає можливість виявляти потенційні загрози в мережевому трафіку та пропонувати шляхи їх усунення. Інтеграція з моделлю ШІ додає цінності, оскільки надає не лише технічний аналіз, але й детальний звіт із рекомендаціями. Гнучкість та інтерактивність інтерфейсу дозволяють користувачам легко отримувати доступ до даних і аналізувати мережеву активність у реальному часі.

Однією з ключових особливостей застосунку є інтеграція з моделлю штучного інтелекту, яка виконує аналіз характеристик трафіку для виявлення потенційних загроз і генерує рекомендації на основі отриманих даних. Це забезпечило адаптивність системи до нових типів загроз, підвищивши її ефективність та надійність. Інтеграція штучного інтелекту дозволила не лише автоматизувати процес виявлення, але й зробити його більш точним та швидким.

Після навчання нейронної мережі ефективність системи значно зросла:

– виявлення зросло з 80% до 97.14%;

– пропущені загрози зменшилися з 450 до 50;

– додаткові можливості поведінкового аналізу забезпечили краще виявлення атак, які не мають явних сигнатур.

Система довела свою ефективність для захисту від більшості тестових мережевих загроз, включаючи реальні сценарії атак, що робить її надійним інструментом для моніторингу та аналізу безпеки мереж.

Для надання повної оцінки розробленої системи її порівняно з іншими існуючими системами моніторингу мережевого трафіку та виявлення загроз. Результати порівняння показали, що розроблена система моніторингу мережевого трафіку демонструє конкурентоспроможність за ключовими показниками, такими як точність виявлення загроз і рівень помилкових спрацьовувань. Порівняно з класичними сигнатурними системами, вона має значно кращі результати у виявленні нових загроз, хоча час обробки трохи вищий. У майбутньому можна працювати над оптимізацією часу обробки для ще більш ефективного використання системи у високонавантажених мережах.

У результаті виконання кваліфікаційної роботи проаналізовано методи аналізу мережевого трафіку та розроблено програмний застосунок, який забезпечує виявлення та блокування зовнішніх вторгнень у комп'ютерні системи за рахунок використання інструментів перехоплення трафіку і аналізуву мережевих пакетів засобами ШІ з метою виявлення та блокування зовнішнього вторгнення.

Для досягнення поставленої мети виконано наступні задачі:

– проведено аналіз існуючих методів моніторингу мережевого трафіку та виявлення вторгнень у комп'ютерні системи, визначено їх переваги та недоліки;

– досліджено сучасні підходи до виявлення аномалій у мережевому трафіку з використанням алгоритмів машинного навчання та визначено найбільш ефективні з них для вирішення завдання виявлення загроз;

– розроблено програмний застосунок моніторингу мережевого трафіку за допомогою базових функцій *Python* з перехоплення мережевих пакетів і *CNN* для визначення аномалій у вмісті мережевих пакетів, що дає можливість виявляти та блокувати зовнішні вторгнення в комп'ютерні системи;

– проведено тестування розробленого програмного застосунку моніторингу мережевого трафіку для виявлення та блокування зовнішнього вторгнення в комп'ютерні системи, що підтвердило доцільність використання розробленого застосунку для вирішення поставленої задачі.

Застосунок поєднує в собі сучасні методи мережевої безпеки, такі як сигнатурний та поведінковий аналіз, а також інтегрує штучний інтелект для автоматизації аналізу загроз і формування рекомендацій для їх усунення. Проведені дослідження та тестування показали, що система здатна ефективно виявляти більшість сучасних загроз, включаючи *DoS*/*DDoS* атаки, незашифровані протоколи та уразливості відкритих портів.

Виявлення вторгнень у комп'ютерні системи ґрунтується на поєднанні кількох підходів, що дозволяє ефективніше реагувати на різні типи загроз. Сигнатурні методи добре працюють для виявлення відомих атак, поведінковий аналіз допомагає

виявляти нові загрози, а кореляція подій і машинне навчання дають змогу виявляти складні атаки. Однак для забезпечення ефективного захисту необхідно використовувати ці методи комплексно, а також постійно оновлювати моделі та алгоритми виявлення, щоб пристосовуватися до нових викликів у сфері кібербезпеки.

Під час аналізу виявлено, що основними проблемами є складність сучасних мереж, обробка великих обсягів даних, висока кількість хибнопозитивних спрацювань, динамічність загроз, шифрування трафіку та захист розподілених мереж. Подолання цих проблем вимагає впровадження інноваційних технологій, таких як машинне навчання, розподілені системи моніторингу та спеціалізовані рішення для *IoT*-пристроїв.

Критерії ефективності систем мережевої безпеки допомагають оцінити надійність, продуктивність і економічну доцільність тих чи інших рішень у сфері кібербезпеки. Для забезпечення належного рівня захисту комп'ютерних мереж важливо, щоб система безпеки була точно налаштованою на виявлення загроз, швидко реагувала на атаки, масштабувалася разом із мережею, мала високу стійкість до збоїв та була зручною для адміністрування.

Проведений аналіз методів моніторингу мережевого трафіку і існуючих програмних рішень, які дозволяють проводити моніторинг мережного трафіку для протидії *DDoS*-атакам, показав, що не дивлячись на велике різноманіття програмних рішень, практично кожне має деякі недоліки. Серед цих недоліків можна виділити складність у налаштуванні, примітивний модуль аналізу трафіку та немоливість зміни алгоритми поведінки даного модулю в залежності від навантаження на сервер.

Примітивні модулі аналізу трафіку часто не здатні розпізнати складні та багаторівневі атаки, що значно знижує їхню корисність у реальних умовах. Це означає, що навіть при наявності захисту від *DDoS*-атак, мережа все одно може бути вразливою до нових і більш витончених методів атак, які не можуть бути виявлені та зупинені за допомогою цих інструментів.

Недостатня можливість змінювати алгоритми поведінки модулів аналізу трафіку в залежності від поточного навантаження на сервер також є значним

недоліком. У випадках високого навантаження, таких як масштабні *DDoS*-атаки, це може призвести до того, що мережевий трафік не буде адекватно оброблений, і сервер буде перевантажений.

Розроблений застосунок продемонстрував високу точність виявлення загроз, яка досягла 97.14% після навчання нейронної мережі, а також значне зниження кількості помилкових спрацьовувань. Це свідчить про надійність системи та її придатність для використання у складних мережевих середовищах. Інтеграція графічного інтерфейсу забезпечила зручність для користувачів, дозволяючи отримувати швидкий доступ до інформації про мережеві пакети та виявлені загрози.

Система є кросплатформною та масштабованою, що дозволяє її впровадження як у невеликих організаціях, так і в корпоративних мережах. Незважаючи на ефективність, у ході роботи було виявлено певні обмеження, такі як збільшення часу обробки через використання нейронних мереж і складність ідентифікації деяких специфічних атак.

Для надання повної оцінки розробленої системи її порівняно з іншими існуючими системами моніторингу мережевого трафіку та виявлення загроз. Результати порівняння показали, що розроблена система моніторингу мережевого трафіку демонструє конкурентоспроможність за ключовими показниками, такими як точність виявлення загроз і рівень помилкових спрацьовувань. Порівняно з класичними сигнатурними системами, вона має значно кращі результати у виявленні нових загроз, хоча час обробки трохи вищий. У майбутньому можна працювати над оптимізацією часу обробки для ще більш ефективного використання системи у високонавантажених мережах.

1. Носок С.О., Фаль О.М., Ткач В.М. Управління інформаційною безпекою: навч. посіб. для студ. спец. 125 «Кібербезпека» / С.О. Носок, О. М. Фаль, В. М. Ткач. – Київ : КПІ ім. Ігоря Сікорського, 2021. 258 с.

2. *Bearden W. O., Ingram T. N., LaForge R.W. Market-Driven Strategies*: *Insights and Innovations / Bearden W. O., Ingram T.N., LaForge R.W. New York*: *Pearson Education*, 2018. 348 *p*.

3. *Taylor R. L. Trends in Customer-Centric Marketing Approaches Across Sectors Journal of Marketing Innovations*. 2020. *Vol*. 45(2). *pp*. 12 28.

4. *Evans D. Advanced Network Security*: *Theories and Applications*. *Boston*: *Springer*, 2020. 350 *p.*

5. *Johansen K. Service Optimization in the Digital Era*: *Marketing Meets Technology London*: *Routledge*, 2023. 648 *p*.

6. *Stevenson A. Cyber Threat Landscape*: *A Holistic Guide to Defense*. *Chicago*: *Pearson Education*, 2021. 260 *p*.

7. *Peterson B. Fundamentals of Secure Network Architecture*. *Berlin*: *Springer*, 2019. 215 *p*.

8. Смирнов А.І. Системи аналізу та виявлення вторгнень у кіберпросторі // Кібербезпека: освіта, наука, техніка. 2021. № 10 (215). С. 56 63.

9. *PCAP Repository*: *The Future of Packet Capture and Analysis* [Електронний ресурс]. Режим доступу: *https*://*pcaptools.org* (Дата звернення: 12.10.2024).

10. *Snort*: *Real-Time Packet Analysis Framework* [Електронний ресурс]. Режим доступу: *https*://*www.snortsec.org* (Дата звернення: 15.10.2024).

11. *Wireshark Essentials for Network Forensics* [Електронний ресурс]. Режим доступу: *https*://*wiresharktools.com* (Дата звернення: 12.10.2024).

12. *Kuznetsov A. A. Telecommunication Systems*: *The Evolution of Infrastructure and Services*. *Kharkiv*: *LLC "Smart Telecom"*, 2022. 1015 *p*.

13. *Serhienko O. P., Lysenko A. M., Mykhalchuk V. S. Multimedia Communication and Networking*: *Theoretical and Practical Insights*. *Kharkiv*: *TechPress*, 2019.

14. *Mockapetris P. Internet Domain Architecture*: *Fundamentals and Advances*. *Network Engineering Group,* 2019. 468 *p*.

15. *Schan C. Workplace Monitoring and Employee Privacy*: *A Modern Dilemma*. *US,* 2022. [Електронний ресурс]. Режим доступу: *https*://*workplace-privacy.net*.

16. *Schan A. Digital Surveillance in Enterprises*: *Case Studies and Trends*. *US,* 2023. [Електронний ресурс]. Режим доступу: *https*://*digitalsecurity.net*.

17. *Taqqu M., Sherman R., Willinger W. Theoretical Foundations of Traffic Behavior in Large-Scale Networks*. *Journal of Advanced Networking*. 2023. *Vol*. 34(3). *pp*. 18 36.

18. *Leland W., Taqqu M., Wilson D. Exploring the Self-Similarity of Modern Network Traffic*. *IEEE Transactions on Communication Systems*. 2023. *Vol*. 31(4). *pp*. 12 30.

19. *Wu H., Kuang, L. Proactive Traffic Engineering in Wide Area Networks Using AI // IEEE Network and Service Management*. 2022. *DOI*: 10.1109/*TNSM*.2022.4483312.

20. *Mendiola A., Astorga J., Higuero M. Emerging Trends in Software-Defined Networking for Traffic Optimization. IEEE Communications Tutorials & Surveys*. 2023. *Vol*. 22(1). *pp*. 865 987.

21. *Johnson M., Peterson L. AI-Driven Solutions for Network Traffic Analysis*. *New York*: *Wiley,* 2025. 280 *p*.

22. *Taylor J. Securing Enterprise Networks*: *Practical Approaches and Challenges*. *San Francisco*: *Packt Publishing,* 2023. 300 *p*.

23. *Brown P., Lewis K. The Rise of Machine Learning in Network Security // Journal of Cybersecurity Research*. 2022. *Vol*. 15(2). *pp*. 45 60.

24. *Wilson, R., Grant L. Real-Time Network Threat Detection Using Behavioral Analysis*. *IEEE Computer Society Journal*. 2023. *Vol*. 29(3). *pp*. 88 103.

25. *IoT Security*: *Analyzing Traffic for Anomalies and Threats // Edited by J. Blackwell*. *Boston*: *MIT Press,* 2025. 320 *p*.

26. Борщ А.А. Методи моніторингу мережевого трафіку з метою виявлення та блокування зовнішнього вторгнення в комп'ютерні системи. Матеріали міжн. наук. інтернет-конф. «Інформаційне суспільство: технологічні, економічні та технічні

аспекти становлення (випуск 92)» (м. Тернопіль, Україна, м. Ополе, Польща, 12-13 листопада 2024 р.). ГО “Наукова спільнота”, *WSZIA w Opolu*. Тернопіль. 2024. *http*://*www.konferenciaonline.org.ua*/*ua*/*article*/*id*-1933/

27. Борщ А.А. Проблеми моніторингу мережевих загроз та запобігання вторгненням в комп’ютерні системи. Тези доповідей міжн. наук.-техн. конф. “Інтелектуальні технології лінгвістичного аналізу” (23-24 жовтня 2024 р.). К.: НАУ, 2024. С. 48.

28. Борщ А.А. Метод моніторингу мережевого трафіку для виявлення зовнішнього вторгнення в комп'ютерну систему. Тези доповідей наук.-практ. конф. “Сучасні тенденції розвитку системного програмування” (21-22 листопада 2023 р.). К.: НАУ, 2024. С. 38-39.

29. *Harris T., Newman G. Distributed Systems Security*: *Monitoring and Prevention Techniques*. *Cambridge*: *Cambridge University Press,* 2024. 290 *p.*

30. Слободян О. Положення про кваліфікаційні роботи (проєкти) здобувачів вищої освіти Національного авіаційного університету. К.: НАУ, 2024. 62 с.

31. ДСТУ 3008-2015 "Інформація та документація. Звіти у сфері науки і техніки. Структура і правила оформлення". К.:ДП "УКРНДНЦ", 2015. 26 с.

Код основного модуля програмного застосунку

```
import ollama
from threading import Event, Thread
from scapy.all import sniff, conf, TCP, UDP, IP, Raw
import tkinter as tk
from tkinter import ttk, messagebox, scrolledtext
import re

analysis_thread = None
terminate_analysis_flag = Event()
ai_model = 'secure-analyzer-v3:latest'

def generate_ai_insight(origin_ip, target_ip, protocol_type, origin_port, target_port, vulnerabilities):
    query = f"Generate a concise response analyzing traffic from {origin_ip} to {target_ip} using {protocol_type} on ports {origin_port}-{target_port}. Identify potential risks and suggest mitigations."
    response = ollama.generate(model=ai_model, prompt=query)
    return response.get("response", "No AI commentary provided")

def fetch_interfaces():
    return [interface.description for interface in conf.ifaces.values()]

def detect_http(traffic_layer):
    if traffic_layer.dport == 80 or traffic_layer.sport == 80:
        return "Detected HTTP traffic over plain text", "Switch to HTTPS"
    return None, None

def detect_syn(traffic_layer):
```

```
        if traffic_layer.flags == 'S':
            return "SYN flag detected, potential flood attack", "Implement SYN flood
countermeasures"
        return None, None

    def detect_credentials(traffic_layer, packet_data):
        if packet_data.haslayer(Raw) and (traffic_layer.dport == 80 or traffic_layer.sport
== 80):
            payload_content = packet_data[Raw].load.decode(errors="ignore")
            if re.search(r'(username=|password=|login=|passwd=)', payload_content,
re.IGNORECASE):
                return f"Unencrypted credentials in payload: {payload_content}", "Encrypt
credentials and use secure communication"
        return None, None

    def detect_ftp(traffic_layer):
        if traffic_layer.dport == 21 or traffic_layer.sport == 21:
            return "Unencrypted FTP session detected", "Utilize FTPS or SFTP protocols"
        return None, None

    def detect_telnet(traffic_layer):
        if traffic_layer.dport == 23 or traffic_layer.sport == 23:
            return "Insecure Telnet connection detected", "Adopt SSH for secure access"
        return None, None

    def inspect_packet(packet_data):
        threats = []
        if packet_data.haslayer(TCP):
            tcp_layer = packet_data.getlayer(TCP)
            checks = [detect_http, detect_syn, detect_credentials, detect_ftp, detect_telnet]
```

```
            for check in checks:
                issue, recommendation = check(tcp_layer) if check != detect_credentials else check(tcp_layer, packet_data)
                if issue:
                    threats.append((issue, recommendation))
        return threats, None

    def initiate_monitoring(interface_name, packet_display, control_button):
        global analysis_thread, terminate_analysis_flag
        terminate_analysis_flag.clear()

        def process_packet(packet_data):
            if not terminate_analysis_flag.is_set() and packet_data.haslayer(IP):
                ip_layer = packet_data[IP]
                source_ip = ip_layer.src
                destination_ip = ip_layer.dst
                protocol = "TCP" if packet_data.haslayer(TCP) else "UDP"

                if packet_data.haslayer(TCP):
                    transport_layer = packet_data[TCP]
                    src_port = transport_layer.sport
                    dest_port = transport_layer.dport
                elif packet_data.haslayer(UDP):
                    transport_layer = packet_data[UDP]
                    src_port = transport_layer.sport
                    dest_port = transport_layer.dport
                else:
                    src_port = "Unknown"
                    dest_port = "Unknown"
```

```
                vulnerabilities, _ = inspect_packet(packet_data)
                commentary = generate_ai_insight(source_ip, destination_ip, protocol,
src_port, dest_port, vulnerabilities)
                if vulnerabilities:
                    for vulnerability, solution in vulnerabilities:
                        row_id = packet_display.insert("", "end", values=(source_ip,
destination_ip, protocol, src_port, dest_port, vulnerability, solution, commentary))
                        packet_display.item(row_id, tags=("highlight",))
                else:
                    row_id = packet_display.insert("", "end", values=(source_ip,
destination_ip, protocol, src_port, dest_port, "No threats identified", "N/A", commentary))

            analysis_thread = Thread(target=sniff, kwargs={"iface": interface_name, "prn":
process_packet, "store": 0})
            analysis_thread.start()
            control_button.config(text="Stop", command=lambda:
terminate_monitoring(control_button, packet_display))

        def terminate_monitoring(control_button, packet_display):
            global terminate_analysis_flag
            terminate_analysis_flag.set()
            control_button.config(text="Start", command=lambda:
initiate_monitoring(interface_name, packet_display, control_button))

        def on_item_selection(event, packet_display):
            try:
                selected_item = packet_display.selection()[0]
                details = packet_display.item(selected_item, "values")
                src_ip, dst_ip, proto, src_port, dst_port, vuln, solution, ai_comment = details
```

```
            info_text = (f"Source IP: {src_ip}\nDestination IP: {dst_ip}\nProtocol:
{proto}\n"
                         f"Source Port: {src_port}\nDestination Port:
{dst_port}\nVulnerability: {vuln}\n"
                         f"Solution: {solution}\nAI Commentary: {ai_comment}")

            detail_window = tk.Tk()
            detail_window.title("Packet Details")
            scroll_text = scrolledtext.ScrolledText(detail_window, width=80, height=15)
            scroll_text.pack(fill='both', expand=True)
            scroll_text.insert('end', info_text)
            scroll_text.config(state='disabled')
            detail_window.mainloop()
        except Exception as error:
            print(f"Error displaying details: {error}")

    def list_interfaces():
        interfaces = fetch_interfaces()
        messagebox.showinfo("Available Interfaces", "\n".join(interfaces))
    def show_threats():
        threat_descriptions = (
            "1. HTTP over plain text: Switch to HTTPS.\n"
            "2. SYN flag detected: Implement SYN flood protection.\n"
            "3. Unencrypted credentials: Use secure protocols.\n"
            "4. Unsecured FTP traffic: Utilize SFTP or FTPS.\n"
            "5. Insecure Telnet traffic: Switch to SSH."
        )
        messagebox.showinfo("Common Threats", threat_descriptions)
    def exit_application(root):
        global terminate_analysis_flag
```

```
        terminate_analysis_flag.set()
        root.destroy()
    def launch_gui(interface_name):
        root = tk.Tk()
        root.title("Traffic Analyzer Tool")
        packet_tree = ttk.Treeview(root, columns=("Source IP", "Destination IP",
"Protocol", "Source Port", "Destination Port", "Vulnerability", "Solution", "AI
Commentary"), show="headings")
        for col in packet_tree["columns"]:
            packet_tree.heading(col, text=col)
        packet_tree.pack(fill="both", expand=True)
        scrollbar = ttk.Scrollbar(root, orient="vertical", command=packet_tree.yview)
        scrollbar.pack(side="right", fill="y")
        packet_tree.configure(yscrollcommand=scrollbar.set)
        control_button = tk.Button(root, text="Start", command=lambda:
initiate_monitoring(interface_name, packet_tree, control_button))
        control_button.pack(side="left", padx=5, pady=5)
        interface_button = tk.Button(root, text="List Interfaces",
command=list_interfaces)
        interface_button.pack(side="left", padx=5, pady=5)
        threat_button = tk.Button(root, text="Show Threats", command=show_threats)
        threat_button.pack(side="left", padx=5, pady=5)
        packet_tree.bind("<Double-1>", lambda event: on_item_selection(event,
packet_tree))
        root.protocol("WM_DELETE_WINDOW", lambda: exit_application(root))
        root.mainloop()
    if __name__ == "__main__":
        interface_name = ""
        launch_gui(interface_name)
```