ДЕРЖАВНИЙ УНІВЕРСИТЕТ ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ КІБЕРБЕЗПЕКИ ТА ЗАХИСТУ ІНФОРМАЦІЇ

КАФЕДРА СИСТЕМ ТА ТЕХНОЛОГІЙ КІБЕРБЕЗПЕКИ

КВАЛІФІКАЦІЙНА РОБОТА

на тему:

«Технологія виявлення аномалій у корпоративній мережі за допомогою IDS з використанням ML»

зі спеціальності *125 Кібербезпека та захист інформації*
(код, найменування спеціальності)

освітньо-професійної програми *Інформаційна та кібернетична безпека*
(назва програми)

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

_____ Андрій МИШКО
(підпис)

Виконав: здобувач вищої освіти групи БСДМ-63
МИШКО Андрій
(прізвище, ім'я)

Керівник д.ф., МАРЧЕНКО Віталій
(науковий ступінь, вчене звання, прізвище, ім'я)

Рецензент _____
(науковий ступінь, вчене звання, прізвище, ім'я)

Київ 2024

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

ML – Machine Learning

IDS – Intrusion Detection System

NIDS – Network-based Intrusion Detection System

HIDS – Host-based Intrusion Detection System

DDoS – Distributed Denial of Service

APT – Advanced Persistent Threat

TCP – Transmission Control Protocol

UDP – User Datagram Protocol

IP – Internet Protocol

ICMP – Internet Control Message Protocol

MAC – Media Access Control

OSI – Open Systems Interconnection

SIEM – Security Information and Event Management

ВСТУП

У сучасному світі швидке зростання обсягів мережевого трафіку та складності кібератак ставлять перед корпоративними мережами нові виклики у сфері кібербезпеки. Традиційні системи виявлення вторгнень (IDS), що базуються на сигнатурному аналізі, стають недостатньо ефективними для протидії сучасним загрозам, які використовують складні та невідомі методи атак. Інтеграція методів машинного навчання з IDS відкриває нові можливості для підвищення ефективності виявлення аномалій у мережевому трафіку. Тому дослідження в області поєднання IDS та машинного навчання є актуальним і відповідає сучасним потребам забезпечення кібербезпеки корпоративних мереж.

Об’єкт дослідження – процеси виявлення аномальної активності у мережевому трафіку корпоративних мереж.

Предмет дослідження – технологія інтеграції систем виявлення вторгнень з методами машинного навчання для підвищення ефективності виявлення аномалій.

Мета роботи – дослідити та розробити технологію виявлення аномалій у корпоративній мережі шляхом інтеграції IDS (на базі Snort) з алгоритмами машинного навчання (Scikit-learn/TensorFlow) для підвищення ефективності виявлення потенційних загроз.

Наукові завдання:

1. Проаналізувати сучасні підходи до виявлення аномалій у мережевому трафіку.

2. Дослідити методи інтеграції Snort із алгоритмами машинного навчання.

3. Розробити модель машинного навчання для класифікації мережевого трафіку.

4. Реалізувати програмне рішення, що інтегрує IDS з моделлю ML.

5. Провести тестування та оцінити ефективність розробленої технології на основі реальних та синтетичних даних.

Методи дослідження – теоретичний аналіз літературних джерел, методи машинного навчання (класифікація, кластеризація), експериментальне моделювання, статистичний аналіз результатів, програмна реалізація на мові Python.

Практичне значення одержаних результатів полягає в розробці технології інтеграції системи виявлення вторгнень Snort із моделями машинного навчання, що дозволяє підвищити ефективність виявлення аномалій у корпоративних мережах та може бути впроваджена в системи кібербезпеки для оперативного реагування на загрози.

Результати даного дослідження доповідались на Всеукраїнській науково-практичній конференції «Актуальні проблеми кібербезпеки» (м. Київ. 24 жовтня 2024).

1 ОГЛЯД СУЧАСНИХ ТЕХНОЛОГІЙ ВИЯВЛЕННЯ АНОМАЛІЙ

1.1. Характеристика систем виявлення вторгнень (IDS), принципи роботи та класифікація

Системи виявлення вторгнень (Intrusion Detection Systems, IDS) є невід'ємною складовою сучасної інфраструктури кібербезпеки, що забезпечує захист корпоративних мереж від несанкціонованого доступу, зловмисних дій та інших кіберзагроз. З розвитком інформаційних технологій та зростанням кількості кібератак, необхідність у використанні сучасних та ефективних рішень стала критичною для підтримки цілісності, конфіденційності та доступності інформаційних ресурсів організацій.

Основна мета IDS полягає у моніторингу та аналізі мережевого трафіку або системних подій з метою виявлення підозрілої активності на основі сигнатур, яка може свідчити про спроби вторгнення або порушення політик безпеки. IDS працюють шляхом збору даних з різних джерел, таких як мережеві пакети, журнали системних подій, файли аудиту та інші, після чого застосовують методи аналізу для виявлення відомих сигнатур атак.

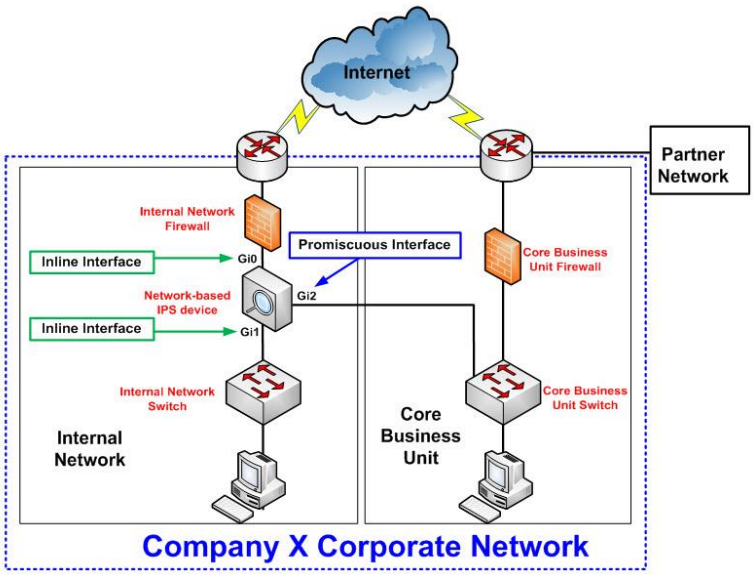

Рис. 1.1. Приклад розгортання IDS

Принцип дії IDS базується на кількох ключових етапах:

Збір даних:

IDS збирає інформацію про мережевий трафік або системні події. У випадку мережевих IDS це здійснюється шляхом перехоплення та аналізу мережевих пакетів, а хостові IDS використовують журнали подій операційної системи, журнали мережевих підключень, журнали роботи мережевих інтерфейсів.

Попередня обробка даних:

Зібрані дані можуть містити непотрібний шум або містити непотрібну інформацію. На цьому етапі відбувається фільтрація, нормалізація та підготовка даних для подальшого аналізу.

Аналіз даних:

IDS застосовує різні методи аналізу для виявлення підозрілої активності. Це може бути порівняння з відомими сигнатурами атак або виявлення аномалій шляхом визначення відхилень від нормальної поведінки.

Виявлення та сповіщення:

У разі виявлення підозрілої активності система генерує сповіщення для адміністратора безпеки або автоматично вживає заходів для запобігання потенційній загрозі.

Системи виявлення вторгнень можна класифікувати за різними ознаками, що відображають їх функціональні можливості та методи роботи. Основні критерії класифікації IDS включають метод аналізу даних та спосіб розгортання системи.

1. Класифікація за методом аналізу даних:

а) Сигнатурні IDS (Signature-Based IDS)

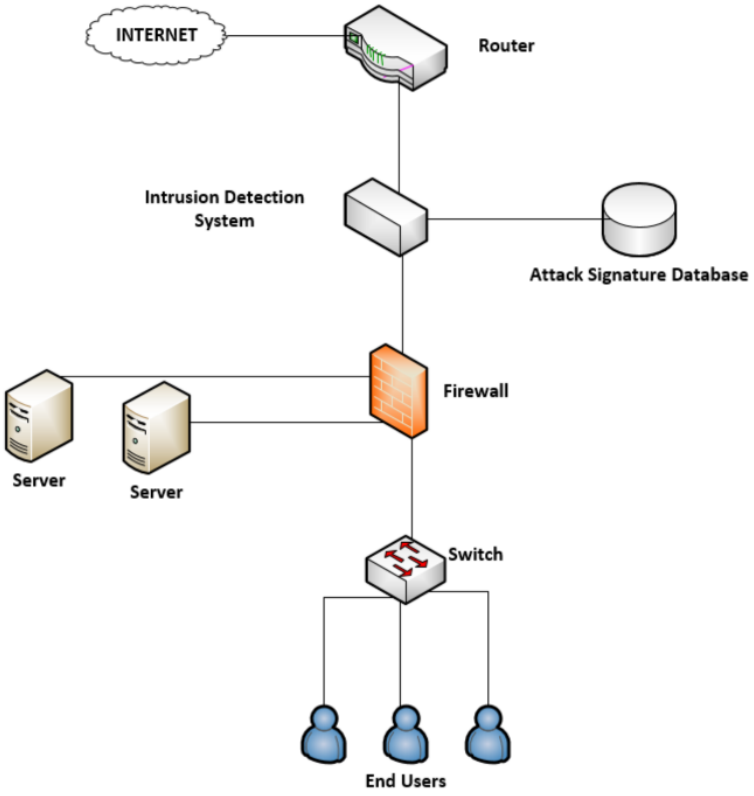

Рис. 1.2. Сигнатурі IDS

Сигнатурні IDS базуються на ідентифікації відомих атак за допомогою попередньо визначених сигнатур. Сигнатура — це унікальний шаблон, який відповідає конкретній атаці або вразливості. Прикладом може бути специфічна послідовність байтів у мережевому пакеті або характерна команда в системному журналі. Система порівнює вхідні дані з базою сигнатур і, при виявленні збігу, генерує сповіщення про можливу атаку. Ефективність сигнатурних IDS підтверджена їх здатністю точно виявляти відомі загрози з мінімальною кількістю хибнопозитивних спрацьовувань. Згідно з дослідженнями, такі системи можуть досягати точності виявлення понад 90% для відомих атак. Вони широко використовуються в корпоративних мережах для забезпечення базового рівня безпеки.

Однак основним недоліком сигнатурних IDS є їх не здатність виявляти нові, невідомі атаки, оскільки для них ще не створено відповідних сигнатур. Це робить їх вразливими до zero-day атак та інших нових загроз. Крім того, ефективність

таких систем залежить від регулярного оновлення бази сигнатур, що вимагає постійної підтримки з боку виробників та адміністраторів системи.

б) Поведінкові IDS (Anomaly-Based IDS)

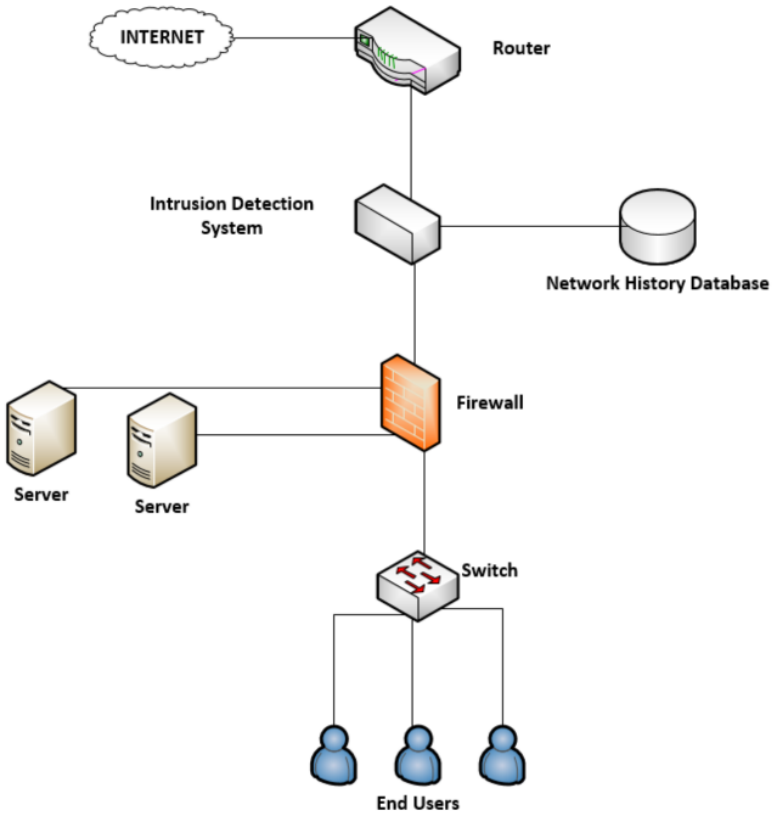

Рис. 1.3. Поведінкові IDS

Поведінкові IDS працюють на основі виявлення аномалій у системній або мережевій активності. Вони створюють модель нормальної поведінки системи, використовуючи методи статистичного аналізу, машинного навчання або нейронних мереж. Будь-яке відхилення від цієї моделі розглядається як потенційна загроза. Поведінкові IDS можуть виявляти невідомі атаки та аномальну активність, яка не відповідає встановленим правилам та відомим паттернам. Це робить їх ефективними проти zero-day атак та інших, складніших загроз. Поведінкові IDS можуть адаптуватися до змін у системі, постійно оновлюючи модель своєї поведінки.

З недоліків, подібні системи часто страждають від високого рівня хибнопозитивних спрацьовувань, оскільки будь-які незвичні, але легітимні дії

можуть бути розцінені як аномалії. Це може створювати додаткове навантаження на адміністраторів безпеки та вимагати постійного налаштування системи для зменшення кількості помилкових сповіщень.

2. Класифікація за способом розгортання:

а) Мережеві IDS (Network-Based IDS, NIDS)

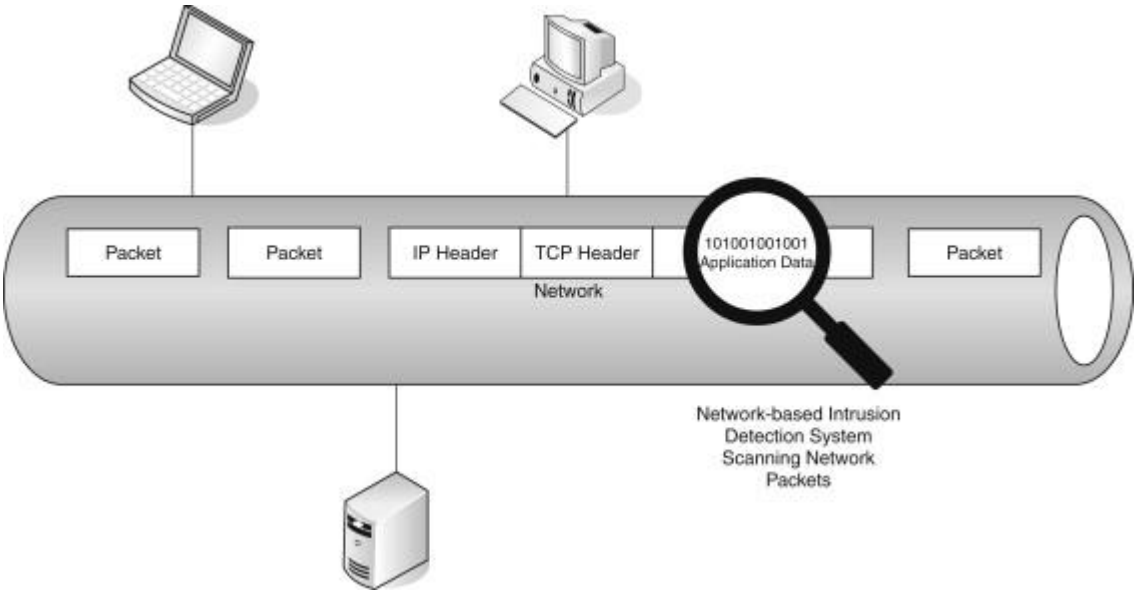

Рис. 1.4. NIDS сканує пакети, що проходять через мережевий інтерфейс.

Мережеві IDS розгортаються в ключових точках мережі, таких як маршрутизатори, комутатори або спеціалізовані пристрої, і аналізують увесь мережевий трафік, що проходить через ці точки. Вони здатні виявляти атаки на рівні мережі, включаючи сканування портів, атаки типу "відмова в обслуговуванні" (DoS/DDoS), спроби проникнення та інші.

NIDS мають перевагу в тому, що вони не вимагають встановлення на кожному окремому вузлі мережі і можуть контролювати трафік між різними сегментами мережі. Дослідження показують, що використання NIDS може знизити ризик успішних мережевих атак на 50%.

Однак мережеві IDS можуть бути обмежені у виявленні атак, що відбуваються всередині зашифрованого трафіку або в межах одного хоста. Крім

того, у високошвидкісних мережах вони можуть не встигати обробляти великий обсяг трафіку в реальному часі, що може призводити до пропуску деяких атак.

б) Хостові IDS (Host-Based IDS, HIDS)

Хостові IDS встановлюються безпосередньо на кінцевих пристроях або серверах і моніторять внутрішню активність системи, включаючи журнали подій, системні файли, реєстр та інші ресурси. Вони здатні виявляти атаки, спрямовані на конкретний хост, такі як спроби ескалації привілеїв, модифікація системних файлів або встановлення шкідливого програмного забезпечення. HIDS надають глибший рівень аналізу подій на хості та можуть виявляти атаки, які не видно на мережевому рівні. Вони особливо ефективні для захисту критичних серверів та систем зберігання даних. Згідно з, впровадження HIDS на ключових вузлах може знизити ймовірність успішних атак на 40%. Недоліком HIDS є те, що вони не можуть контролювати мережевий трафік між хостами та вимагають встановлення та налаштування на кожному окремому пристрої. Це може бути складним у великих мережах з великою кількістю кінцевих точок.

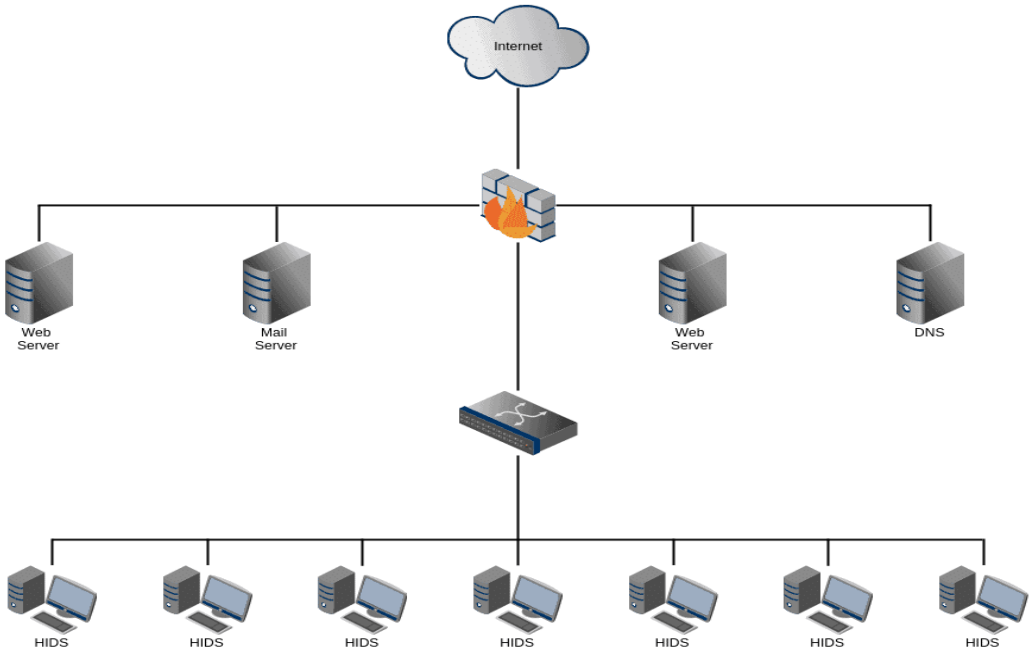

Рис. 1.5. Схема розгортання. HIDS встановлюється на кожен хост, тоді як NIDS розміщується в мережі

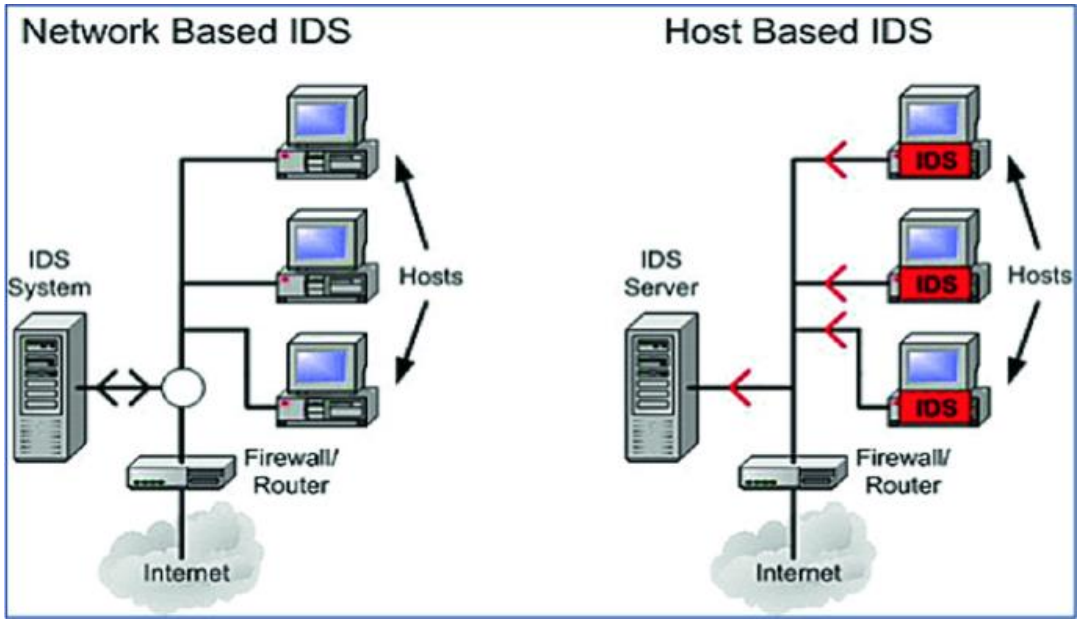

Рис. 1.7. Схематичне порівняння мережевої та хостової систем виявлення вторгнень.

У практиці часто використовуються комбіновані рішення, де сигнатурні IDS забезпечують швидке виявлення відомих атак, а поведінкові IDS доповнюють їх, виявляючи нові та невідомі загрози. Наприклад, система Snort, яка є сигнатурною NIDS, може бути інтегрована з модулями машинного навчання для впровадження поведінкового аналізу.

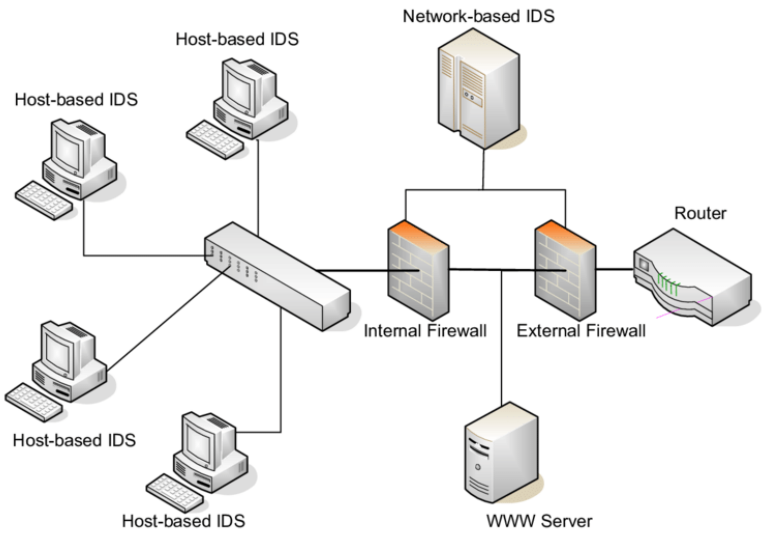

Рис. 1.8. Приклад імплементації NIDS та HIDS

1.2. Використання методів машинного навчання у системах IDS

У сучасних системах виявлення вторгнень (IDS) інтеграція методів машинного навчання (ML) стає дедалі більш актуальною через необхідність ефективного розпізнавання складних та динамічних кіберзагроз. Традиційні підходи до IDS, зокрема сигнатурні та поведінкові методи, мають свої обмеження, зокрема обмежену здатність до адаптації до нових атак та високу ймовірність хибнопозитивних спрацьовувань. Методи ML, завдяки своїй здатності до навчання на основі великих обсягів даних та виявлення прихованих закономірностей, пропонують значні переваги в покращенні точності та ефективності систем IDS.

Машинне навчання дозволяє автоматизувати процес аналізу мережевого трафіку, зменшуючи залежність від ручного оновлення сигнатур та правил. Це особливо важливо в умовах постійно змінюваних загроз, де швидкість реакції на нові види атак є критичною для забезпечення безпеки мережі. Завдяки алгоритмам класифікації та кластеризації, ML-моделі здатні виявляти як відомі, так і невідомі загрози, аналізуючи різні характеристики мережевого трафіку, такі як IP-адреси, порти, протоколи, обсяг даних та поведінкові патерни.

Використання ML у IDS дає можливість побудови адаптивних моделей, які самостійно оновлюються у відповідь на нові дані. Наприклад, алгоритми ансамблевих методів, такі як Random Forest та Gradient Boosting, демонструють високу точність у класифікації трафіку, поєднуючи декілька слабких моделей для створення потужнішої системи виявлення. Ці методи здатні ефективно працювати з нерівномірно розподіленими даними та мають стійкість до переобучення, що робить їх придатними для застосування в реальних мережевих середовищах.

Нейронні мережі, зокрема глибокі навчальні моделі, відкривають нові можливості для аналізу складних структур даних та виявлення нелінійних взаємозв'язків між різними параметрами мережевого трафіку. Використання рекурентних нейронних мереж (RNN) дозволяє ефективно аналізувати часові залежності у трафіку, що є важливим для виявлення тривалих атак, таких як

Advanced Persistent Threats (APT). Автокодувальники, як ще один приклад глибоких моделей, використовуються для виявлення аномалій шляхом відновлення вхідних даних і аналізу помилок реконструкції, що може свідчити про наявність невідомих загроз.

Ефективність методів машинного навчання у системах IDS підтверджена численними дослідженнями. Наприклад, дослідження Sommer & Paxson, 2010 показало, що використання алгоритмів машинного навчання дозволяє значно покращити точність виявлення атак порівняно з традиційними сигнатурними методами. Інші роботи демонструють, що комбінування різних ML-алгоритмів може забезпечити більш високу стійкість до різноманітних типів атак, що підвищує загальну ефективність системи безпеки.

Актуальним є питання інтеграції ML-моделей з існуючими системами IDS. Це включає не лише технічні аспекти, такі як обробка великих обсягів даних та забезпечення реального часу аналізу, але й організаційні фактори, зокрема необхідність у висококваліфікованих спеціалістах для налаштування та підтримки таких систем. Впровадження ML у IDS вимагає ретельного планування, вибору відповідних алгоритмів та забезпечення достатньої кількості навчальних даних для досягнення високої точності та надійності виявлення загроз.

Використання методів машинного навчання у системах IDS представляє собою перспективний напрямок розвитку кібербезпеки, що дозволяє подолати обмеження традиційних підходів та забезпечити більш високий рівень захисту корпоративних мереж. Подальші дослідження в цій галузі спрямовані на вдосконалення алгоритмів, підвищення їхньої адаптивності та зменшення кількості хибнопозитивних спрацьовувань, що є ключовими чинниками для успішного впровадження ML у практичні системи безпеки.

1.3. Аналіз загроз і сценаріїв атак при використанні IDS з машинним навчанням у корпоративних мережах

Сучасні корпоративні мережі стикаються з різноманітними та постійно еволюціонуючими кіберзагрозами, що вимагають впровадження ефективних систем захисту. Інтеграція систем виявлення вторгнень з методами машинного навчання стає ключовим елементом у забезпеченні безпеки, дозволяючи організаціям оперативно реагувати на складні та невідомі атаки. Аналіз загроз та сценаріїв атак є невід'ємною частиною розробки та вдосконалення таких систем, оскільки дозволяє точно налаштовувати алгоритми ML для ефективного виявлення аномалій у мережевому трафіку.

В даному розділі є Advanced Persistent Threats. Ці загрози характеризуються високою складністю, тривалістю та цілеспрямованістю. Зловмисники, що використовують APT, часто мають значні ресурси та знання про цільову інфраструктуру, що дозволяє їм обходити традиційні засоби захисту. Наприклад, атака на компанію Sony Pictures у 2014 році демонструє, як APT може призвести до значних фінансових збитків та репутаційних втрат. Використання ML у IDS дозволяє аналізувати великі обсяги даних та виявляти тонкі відхилення у поведінці користувачів та пристроїв, що можуть свідчити про початок або перебіг APT-атаки. Алгоритми класифікації, такі як Random Forest та Support Vector Machine, можуть ефективно ідентифікувати нетипові патерни активності, які часто супроводжують APT.

Значущою загрозою є і Distributed Denial of Service (DDoS) атаки, які спрямовані на виснаження ресурсів мережі або сервісів, роблячи їх недоступними для користувачів. У 2020 році атака на сервіс Dyn, яка паралізувала доступ до популярних веб-сайтів, показала, наскільки серйозними можуть бути наслідки DDoS-атак. Використання ML у IDS дозволяє оперативно визначати аномальні піки в трафіку та автоматично реагувати на них. Алгоритми кластеризації, такі як K-Means та DBSCAN, можуть виявляти масові спроби доступу до мережі, що

характерно для DDoS-атаки, та запускати механізми фільтрації або блокування підозрілих джерел трафіку.

Malware або шкідливе програмне забезпечення є ще однією поширеною загрозою для корпоративних мереж. Віруси, трояни, руткіти та інші види малваре можуть порушувати цілісність систем, красти конфіденційні дані або забезпечувати зворотний доступ до зловмисників. Наприклад, атака WannaCry у 2017 році поширилася на сотні тисяч комп'ютерів по всьому світу, завдаючи великих збитків бізнесу та інфраструктурі. Інтеграція ML у IDS дозволяє виявляти невідомі варіанти малваре, аналізуючи поведінкові патерни програм, які відрізняються від звичайних операцій. Нейронні мережі, зокрема глибокі автокодувальники, можуть аналізувати поведінку програмного забезпечення та визначати аномалії, що вказують на наявність малваре.

Фішингові атаки також залишаються однією з найпоширеніших форм соціальної інженерії, спрямованих на обман користувачів з метою викрадення облікових даних або фінансових ресурсів. Атаки типу "spear-phishing", що націлені на конкретних осіб або організації, можуть бути особливо ефективними. Використання ML у IDS дозволяє аналізувати електронні листи та мережеві запити на предмет схожості з відомими фішинговими схемами або виявляти нові, невідомі патерни, що характеризують фішингові спроби. Це забезпечує додатковий рівень захисту, дозволяючи зменшити кількість успішних фішингових атак.

Експлойти та zero-day атаки також становлять значну загрозу для корпоративних мереж. Zero-day атаки використовують невідомі раніше вразливості в програмному забезпеченні, що робить їх особливо небезпечними, оскільки традиційні сигнатурні системи не можуть їх виявити до випуску патчів. Інтеграція ML у IDS дозволяє аналізувати поведінку програмного забезпечення та мережевий трафік, визначаючи невідомі патерни, що можуть свідчити про спробу використання zero-day вразливостей. Це робить можливим виявлення атак до того, як вони можуть завдати шкоди організації. Internal threats або внутрішні загрози,

що включають дії співробітників або зловмисників, які мають доступ до внутрішніх ресурсів організації, також можуть бути ефективно виявлені за допомогою ML.

Внутрішні загрози можуть включати несанкціонований доступ до конфіденційних даних, спроби порушення політик безпеки або зловживання привілеями. Аналіз поведінки користувачів та пристроїв за допомогою ML дозволяє виявляти нетипову активність, яка може свідчити про внутрішню загрозу. Алгоритми аномалій, такі як One-Class SVM або Isolation Forest, можуть ідентифікувати нетипові патерни доступу до даних або змін у системних налаштуваннях, що вказують на потенційну загрозу.

Крім зазначених загроз, сучасні корпоративні мережі також піддаються атакам типу Man-in-the-Middle (MitM), SQL-ін’єкціям та іншим формам web-атак, які можуть призвести до витоку даних або порушення роботи систем. Використання ML у IDS дозволяє автоматизувати процес виявлення таких атак шляхом аналізу мережевих запитів та визначення аномалій у взаємодії між клієнтами та серверами. Це забезпечує своєчасну реакцію на спроби маніпуляції даними або порушення цілісності систем.

Впровадження ML у IDS не лише підвищує здатність системи виявляти різноманітні загрози, але й забезпечує гнучкість у адаптації до нових та змінних умов мережевого середовища. Це дозволяє корпоративним мережам ефективно протидіяти сучасним кіберзагрозам, зменшуючи ризик успішних атак та забезпечуючи безпеку критично важливих інформаційних ресурсів.

Впровадження ML у IDS дозволяє значно покращити здатність мереж до виявлення складних атак та адаптації до нових загроз, що є критично важливим у сучасному динамічному середовищі кібербезпеки. Наступний розділ буде присвячений теоретичним аспектам машинного навчання, включаючи основи, класифікацію алгоритмів та огляд бібліотек Python, що є фундаментальними для розробки та реалізації ефективних моделей виявлення аномалій.

2 ТЕОРЕТИЧНІ АСПЕКТИ МАШИННОГО НАВЧАННЯ

2.1. Основи машинного навчання, класифікація, алгоритми та моделі

Машинне навчання (Machine Learning, ML) представляє собою галузь штучного інтелекту, яка зосереджена на розробці алгоритмів і моделей, здатних самостійно покращувати свої характеристики шляхом аналізу даних. На відміну від традиційного програмування, де правила та логіка визначаються вручну, машинне навчання дозволяє системам вчитися на основі вхідних даних і досвіду, що значно розширює їхні можливості виявлення складних закономірностей та адаптації до нових умов.

Класифікація є одним з ключових напрямків машинного навчання, спрямованим на розподіл об'єктів або подій у визначені категорії на основі їхніх ознак. У контексті виявлення аномалій у мережевому трафіку класифікаційні моделі використовуються для розрізнення нормального трафіку від потенційно шкідливого. Основними типами класифікації є бінарна класифікація, яка передбачає розподіл об'єктів на дві категорії, та багатокласова класифікація, де об'єкти можуть належати до однієї з декількох категорій.

Алгоритми класифікації варіюються за складністю та методами роботи. Однією з найбільш популярних є метод опорних векторів (Support Vector Machine, SVM), який прагне знайти гіперплощину, що найкраще розділяє різні класи в багатовимірному просторі. Цей підхід ефективний при роботі з високорозмірними даними і демонструє високу точність у задачах розпізнавання образів та класифікації тексту.

Дерева рішень (Decision Trees) є іншим широко використовуваним алгоритмом, що працює шляхом побудови ієрархічної структури, де кожен вузол представляє собою рішення, а гілки ведуть до наступних рішень або кінцевих результатів. Цей метод інтуїтивно зрозумілий та легкий у візуалізації, однак може бути схильним до переобучення, якщо не застосовуються відповідні методи регуляризації.

Випадкові ліси (Random Forest) розширюють концепцію дерев рішень, створюючи ансамбль з багатьох дерев, кожне з яких навчається на випадковій підмножині даних та ознак. Комбінація прогнозів усіх дерев дозволяє підвищити загальну точність моделі та знизити ризик переобучення, що робить цей метод особливо корисним для складних задач класифікації.

Нейронні мережі, зокрема глибокі нейронні мережі (Deep Neural Networks), забезпечують потужний інструмент для моделювання складних нелінійних залежностей у даних. Завдяки багаторівневій структурі, нейронні мережі здатні автоматично витягувати високорівневі ознаки з сирих даних, що робить їх ефективними для задач візуального розпізнавання, обробки природної мови та інших складних задач.

Побудова моделей машинного навчання включає кілька етапів, починаючи від збору та підготовки даних до вибору відповідного алгоритму та його налаштування. Важливою складовою цього процесу є вибір правильних ознак (features), які найбільш релевантні для задачі класифікації. Наприклад, у випадку аналізу мережевого трафіку такими ознаками можуть бути IP-адреси джерела та призначення, протоколи, розмір пакетів та час між пакетами.

Навчання моделі передбачає оптимізацію її параметрів на основі навчального набору даних, щоб мінімізувати помилки класифікації. Після навчання модель проходить стадію тестування на незалежному наборі даних, що дозволяє оцінити її здатність до узагальнення та виявлення аномалій у нових даних. Оцінка якості моделі здійснюється за допомогою різних метрик, таких як точність (accuracy), повнота (recall), точність виявлення (precision) та F1-міра, які дозволяють комплексно оцінити її ефективність.

Застосування методів машинного навчання у системах IDS забезпечує можливість більш гнучкого та адаптивного підходу до виявлення загроз. Моделі ML можуть постійно оновлюватися та адаптуватися до нових видів атак, що значно підвищує їхню ефективність порівняно з традиційними сигнатурними методами. Крім того, використання алгоритмів класифікації та кластеризації дозволяє не лише

виявляти відомі загрози, але й ідентифікувати нові, раніше невідомі види атак, що робить системи IDS більш стійкими до динамічно змінних умов кібербезпеки.

Моделі машинного навчання, завдяки своїй здатності до самонавчання та адаптації, стають невід'ємною частиною сучасних систем кібербезпеки, забезпечуючи ефективний захист корпоративних мереж від різноманітних форм кіберзагроз.

2.2. Огляд Python-бібліотек для побудови моделей та тестування

Мова програмування Python займає провідне місце у сфері машинного навчання та аналізу даних завдяки своїй гнучкості, простоті використання та широкому спектру доступних бібліотек. У контексті розробки систем виявлення аномалій у корпоративних мережах, Python забезпечує ефективні інструменти для обробки даних, побудови моделей машинного навчання та їх тестування, що робить її ідеальним вибором для цієї роботи.

Однією з ключових бібліотек, що використовується у проекті, є Pandas. Ця бібліотека забезпечує потужні можливості для маніпуляції даними, дозволяючи ефективно завантажувати, очищувати та трансформувати великі набори даних. Pandas дозволяє легко працювати з структурованими даними, що особливо важливо при обробці логів Snort, де необхідно швидко виділяти релевантні ознаки та готувати їх для подальшого аналізу.

NumPy є ще однією незамінною бібліотекою, яка забезпечує підтримку великих багатовимірних масивів та матриць, а також надає широкий набір математичних функцій для ефективних обчислень. У поєднанні з Pandas, NumPy дозволяє здійснювати високопродуктивні обчислювальні операції, що необхідні для підготовки даних перед навчанням моделей машинного навчання.

Для побудови моделей машинного навчання використовується Scikit-learn. Ця бібліотека пропонує широкий спектр алгоритмів для класифікації, регресії, кластеризації та зменшення розмірності, що робить її універсальним інструментом для вирішення різноманітних задач аналізу даних. Scikit-learn інтегрується з Pandas

та NumPy, забезпечуючи безперебійну роботу з даними і дозволяючи швидко розробляти та тестувати різні моделі класифікації для виявлення аномалій у мережевому трафіку.

У випадку застосування глибокого навчання, TensorFlow стає ключовим інструментом завдяки своїй здатності створювати складні нейронні мережі та ефективно їх тренувати на великих наборах даних. TensorFlow підтримує автоматичне диференціювання та оптимізацію моделей, що є критично важливим для побудови точних та ефективних моделей виявлення аномалій. Інтеграція з Keras, високорівневою API, спрощує процес створення та налаштування глибоких нейронних мереж, дозволяючи швидко експериментувати з різними архітектурами та параметрами моделей.

Вибір Python для реалізації систем виявлення аномалій обумовлений кількома факторами. По-перше, Python має великий вибір бібліотек, спеціально розроблених для машинного навчання та аналізу даних, що значно спрощує процес розробки та інтеграції моделей. По-друге, Python забезпечує високий рівень інтеграції з іншими інструментами та системами, такими як Snort, що дозволяє створювати комплексні рішення для кібербезпеки.

2.3. Особливості мережевого трафіку, обробка та формування ознак

Мережевий трафік характеризується високою динамічністю та різноманітністю, що робить його аналіз для виявлення аномалій складним завданням. Основні особливості мережевого трафіку включають велике обсяги даних, високу швидкість передачі інформації, багатогранність протоколів та різні типи взаємодій між пристроями. Ці характеристики вимагають ефективних методів обробки та формування ознак, які можуть адекватно відображати поведінку мережі для подальшого аналізу машинним навчанням.

Обробка мережевого трафіку зазвичай починається з збору даних за допомогою систем виявлення вторгнень (IDS), таких як Snort або Suricata. Зібраний трафік містить різноманітні інформаційні поля, включаючи IP-адреси джерела та

призначення, порти, протоколи, розміри пакетів, час між пакетами та інші метрики. Для подальшого аналізу ці дані повинні бути очищені від шумів та непотрібної інформації. Це включає видалення дублікатів, заповнення пропущених значень та нормалізацію даних, що є критично важливим для забезпечення якості навчального набору.

Формування ознак полягає у виборі та трансформації релевантних атрибутів з сирих даних трафіку. Основним завданням цього етапу є виявлення таких характеристик, які найбільш інформативні для виявлення аномалій. Наприклад, аналіз частоти з'єднань з певними IP-адресами, середній розмір пакетів, розподіл часу між пакетами та використання протоколів може надати важливу інформацію про поведінку мережі. Важливим аспектом є також кодування категоріальних змінних, таких як протоколи або типи атак, у числовий формат, що дозволяє моделям машинного навчання ефективно їх обробляти.

Масштабування даних є необхідним кроком для забезпечення того, щоб всі ознаки мали однаковий масштаб, що запобігає домінуванню одних ознак над іншими під час навчання моделі. Найпоширенішими методами масштабування є стандартизація, яка перетворює дані так, щоб вони мали середнє значення 0 і стандартне відхилення 1, та нормалізація, яка масштабувала дані до певного діапазону, зазвичай від 0 до 1. Ці методи сприяють покращенню конвергенції алгоритмів навчання та підвищують стабільність моделей.

Крім того, процес формування ознак часто включає виділення високорівневих характеристик за допомогою методів зменшення розмірності, таких як головні компоненти аналізу (PCA) або методи вибору ознак. Це дозволяє зменшити кількість ознак, зберігаючи при цьому найбільш інформативні аспекти даних, що сприяє покращенню продуктивності та зменшенню ризику переобучення моделей.

Обробка мережевого трафіку включає балансування даних, оскільки в реальних умовах аномалій у трафіку зазвичай значно менше, ніж нормальної активності. Використання методів балансування, таких як oversampling або undersampling, допомагає створити більш збалансовані навчальні набори, що

покращує здатність моделей до виявлення рідкісних аномалій без збільшення кількості хибнопозитивних спрацьовувань.

Важливою складовою є побудова інформативних метрик для оцінки якості моделей машинного навчання. Метрики, такі як точність (accuracy), повнота (recall), точність виявлення (precision) та F1-міра, дозволяють комплексно оцінити ефективність моделі виявлення аномалій, забезпечуючи баланс між мінімізацією хибнопозитивних та хибно негативних спрацьовувань.

Особливістю мережевого трафіку є його постійна еволюція, що вимагає регулярного оновлення моделей та адаптації до нових типів атак. Це означає, що процес обробки даних та формування ознак повинен бути автоматизованим та інтегрованим з системами безпеки, що дозволяє оперативно реагувати на зміни в мережевій поведінці та забезпечувати актуальність моделей машинного навчання.

3 РОЗРОБКА ТА РЕАЛІЗАЦІЯ ТЕХНОЛОГІЇ ВИЯВЛЕННЯ АНОМАЛІЙ

3.1. Вибір і опис середовища розробки

У даній роботі було обрано мову програмування Python завдяки її численним перевагам, що роблять її ідеальним інструментом для реалізації комплексних систем кібербезпеки.

Python відзначається високим рівнем читабельності та простотою синтаксису, що значно спрощує процес розробки та обслуговування коду. Це особливо важливо у контексті проекту, де необхідно швидко інтегрувати різні компоненти, такі як система Snort та моделі машинного навчання. Завдяки зрозумілому синтаксису, програмісти можуть ефективно співпрацювати, зменшуючи час на написання та налагодження коду.

Дана мова програмування має широкий спектр спеціалізованих бібліотек та фреймворків, які полегшують процес розробки моделей машинного навчання. Бібліотеки Pandas та NumPy забезпечують великий інструментарій для маніпуляції даними, дозволяючи ефективно виконувати операції з великими наборами даних, що характерні для мережевого трафіку. Scikit-learn пропонує різноманітні алгоритми машинного навчання, що дозволяють швидко створювати та тестувати моделі класифікації та кластеризації, необхідні для виявлення аномалій.

Python має значні інтеграційні можливості з іншими системами та інструментами. У даному проекті необхідно інтегрувати Snort із Python для обробки логів та передачі даних до моделей машинного навчання. Python забезпечує безперешкодну взаємодію з системами через API та бібліотеки для роботи з файлами, що дозволяє створювати гнучкі та масштабовані рішення.

Додатково, він підтримує інтерактивні середовища розробки, такі як Jupyter Notebook, що сприяє експериментальному підходу до розробки та тестування моделей. Це дозволяє дослідникам швидко ітеративно розробляти та оцінювати різні алгоритми машинного навчання, адаптуючи їх до специфічних вимог

Для реалізації технології виявлення аномалій було обрано набір інструментів та бібліотек, основними інструментами є:

Snort — вибрано як базову систему IDS завдяки її високій надійності, гнучкості налаштувань та широкій підтримці спільноти. Snort дозволяє детально моніторити мережевий трафік та генерувати структуровані логи, які використовуються для навчання моделей машинного навчання.

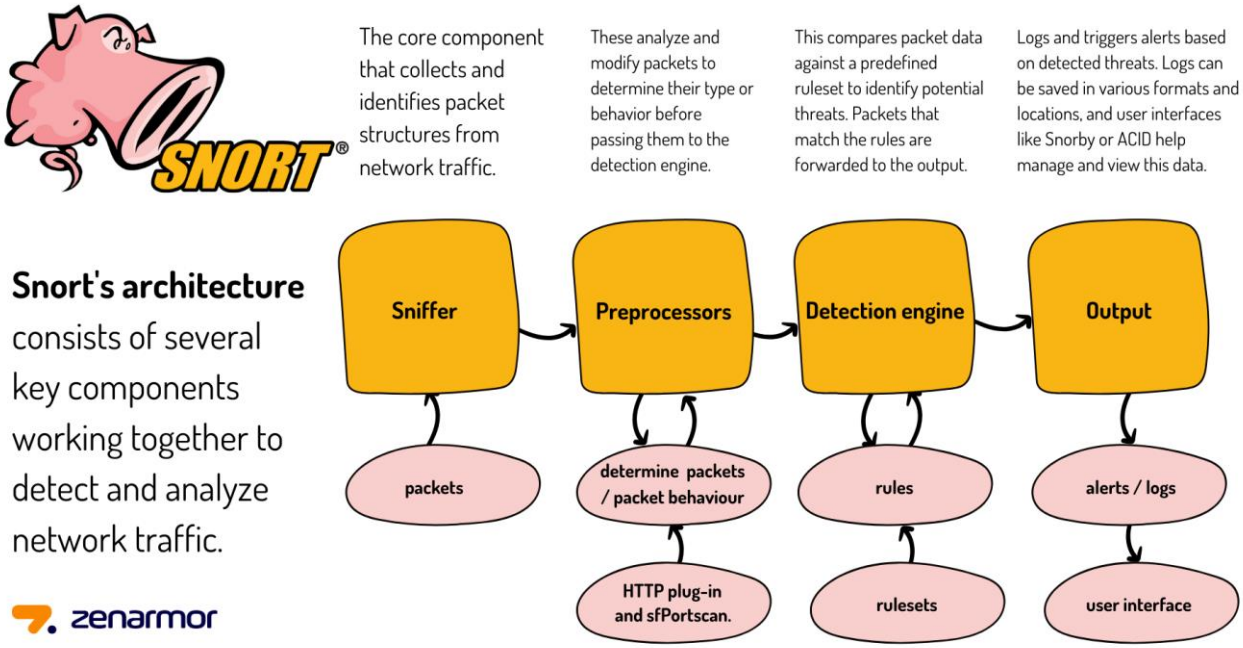

Рис. 3.1. Архітектура IDS Snort

Python — обрана мова програмування забезпечує необхідну гнучкість та доступ до широкого спектру бібліотек для обробки даних та машинного навчання.

Рис. 3.2. Мова програмування Python

Pandas — бібліотека мови програмування Python, використовується для ефективної обробки та аналізу великих наборів даних, що генеруються Snort. Pandas дозволяє легко маніпулювати даними, виконувати очищення та трансформацію для подальшого аналізу.

Рис. 3.3. Бібліотека Python Pandas

Scikit-learn — бібліотека для побудови та навчання моделей машинного навчання завдяки її простоті використання та широкому набору алгоритмів класифікації та кластеризації.

Рис. 3.4. Бібліотека Python Scikit Learn

TensorFlow — використовується для розробки глибоких нейронних мереж, що дозволяють виявляти складні патерни у мережевому трафіку. Забезпечує високу продуктивність та масштабованість моделей, що необхідно для обробки великих обсягів даних у реальному часі.

Рис. 3.5. Бібліотека Python TensorFlow

Для ефективної розробки та реалізації системи було обрано середовище розробки PyCharm. PyCharm відзначається високою продуктивністю, зручним інтерфейсом та потужними можливостями.

Рис. 3.6. Середовище розробки (IDE) PyCharm.

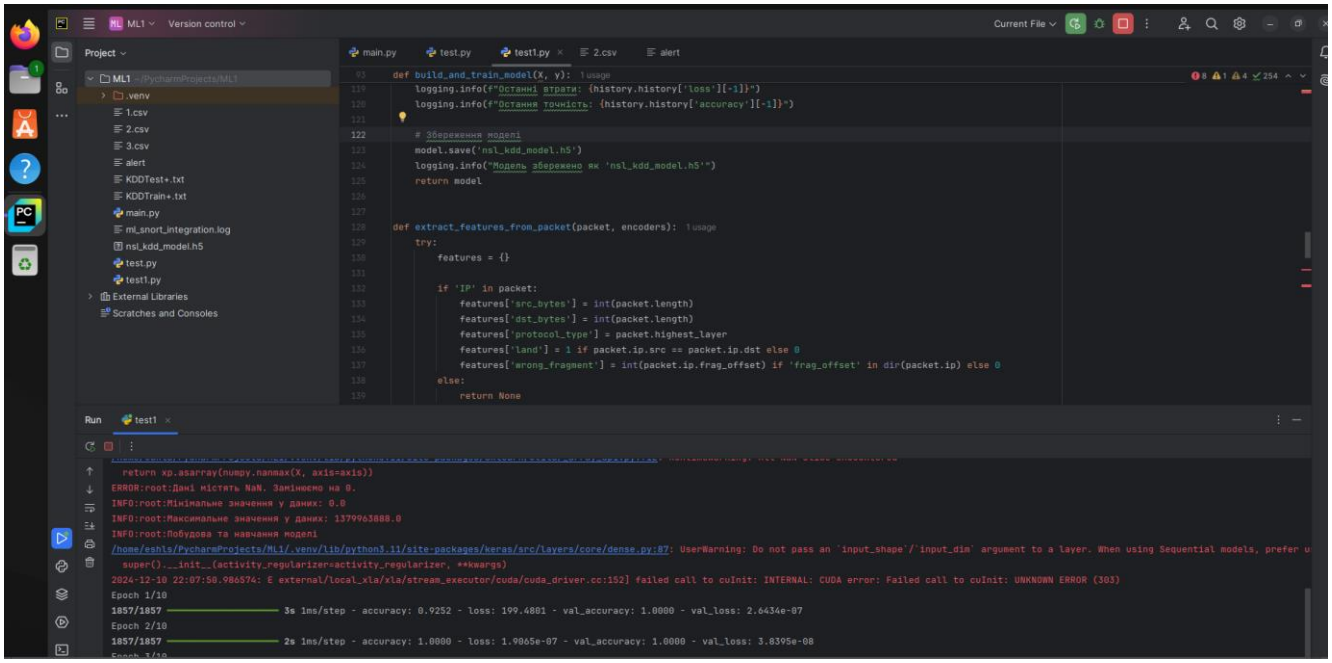

Рис. 3.7. Інтерфейс PyCharm

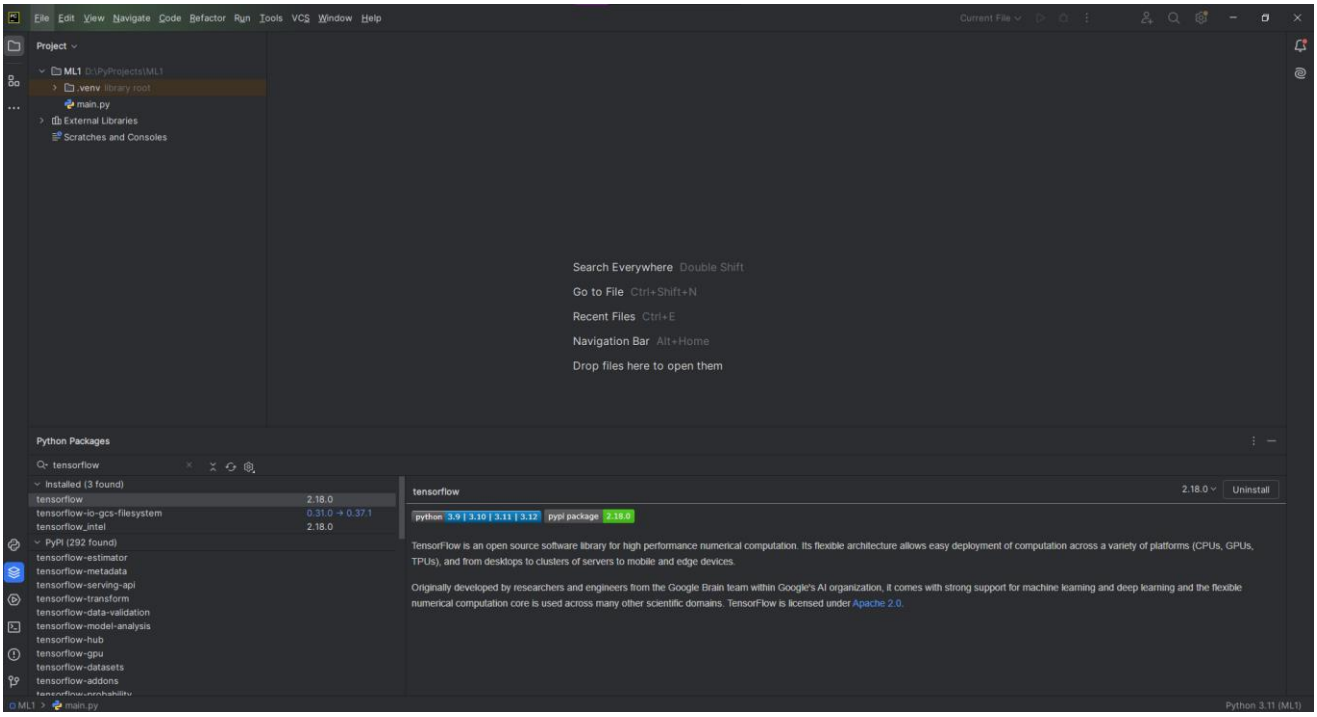

Рис. 3.8. Інтерфейс PyCharm

3.2. Побудова ML-моделі, інтеграція Snort, тестування та оцінка

Першим етапом було встановлення всіх необхідних бібліотек для розробки та навчання моделей машинного навчання. Для цього було використано менеджер пакетів pip, який дозволяє встановлювати та управляти бібліотеками Python.

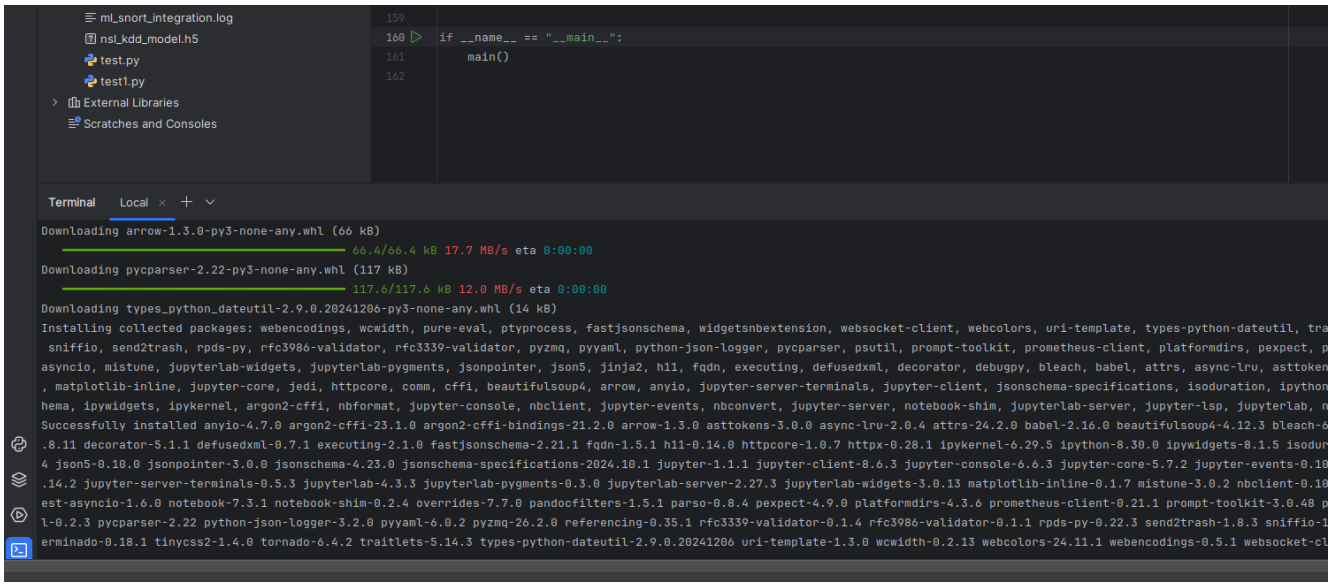

Рис. 3.9. Встановлення необхідних бібліотек та залежностей

Наступним кроком було встановлення системи виявлення вторгнень Snort, яка служить основою для збору та аналізу мережевого трафіку. Для встановлення Snort було обрано операційну систему Ubuntu, оскільки вона забезпечує стабільну та підтримувану платформу для запуску Snort.

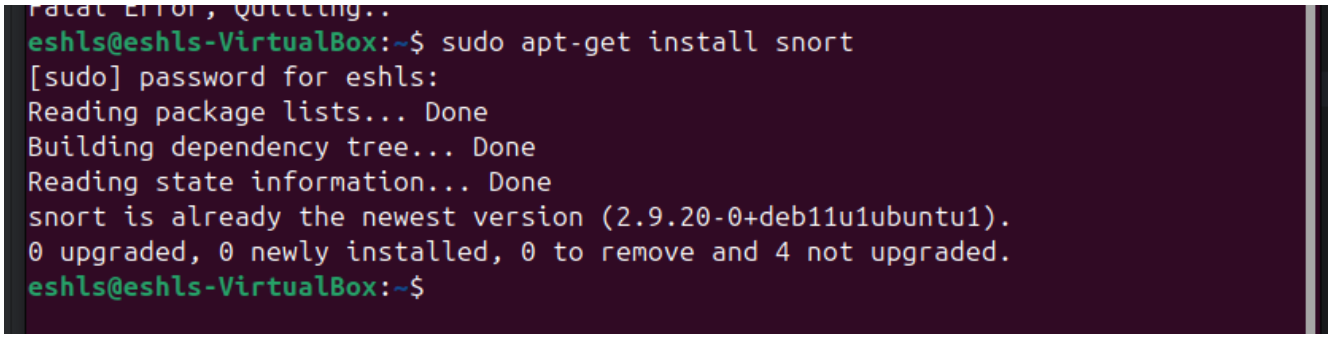

Рис. 3.10. Встановлення Snort

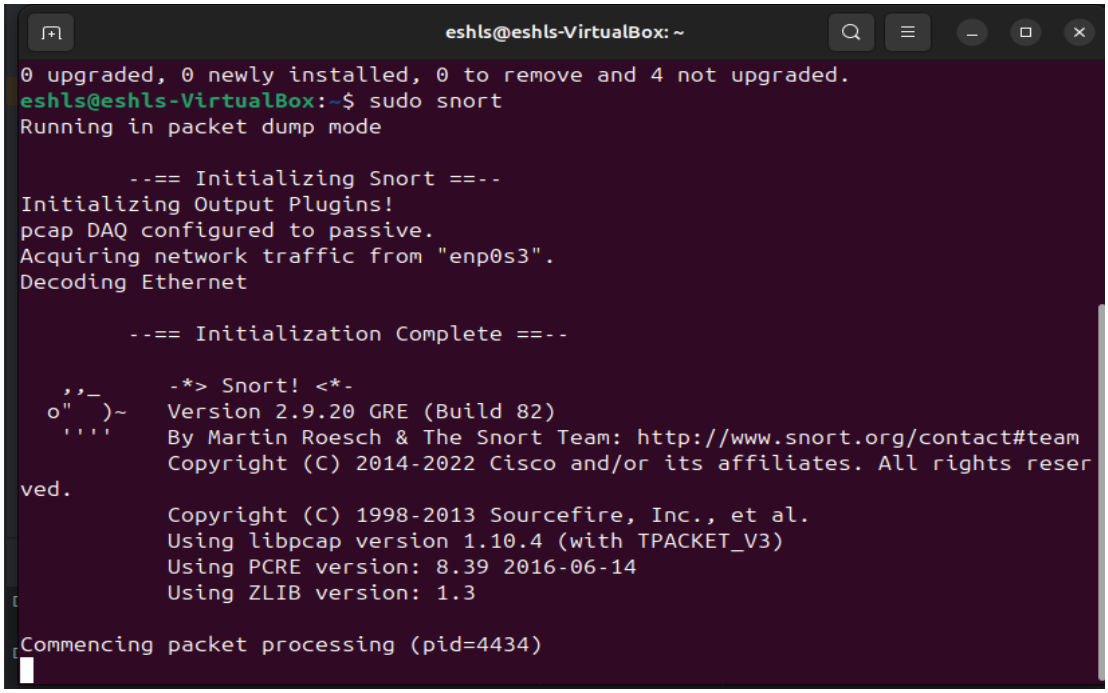

Рис. 3.11. Ініціалізація Snort

Після успішної установки Snort необхідно налаштувати конфігураційний файл для моніторингу потрібного мережевого інтерфейсу та визначення мережевого діапазону, який буде захищатися. Файл конфігурації знаходиться за шляхом /etc/snort/snort.conf.

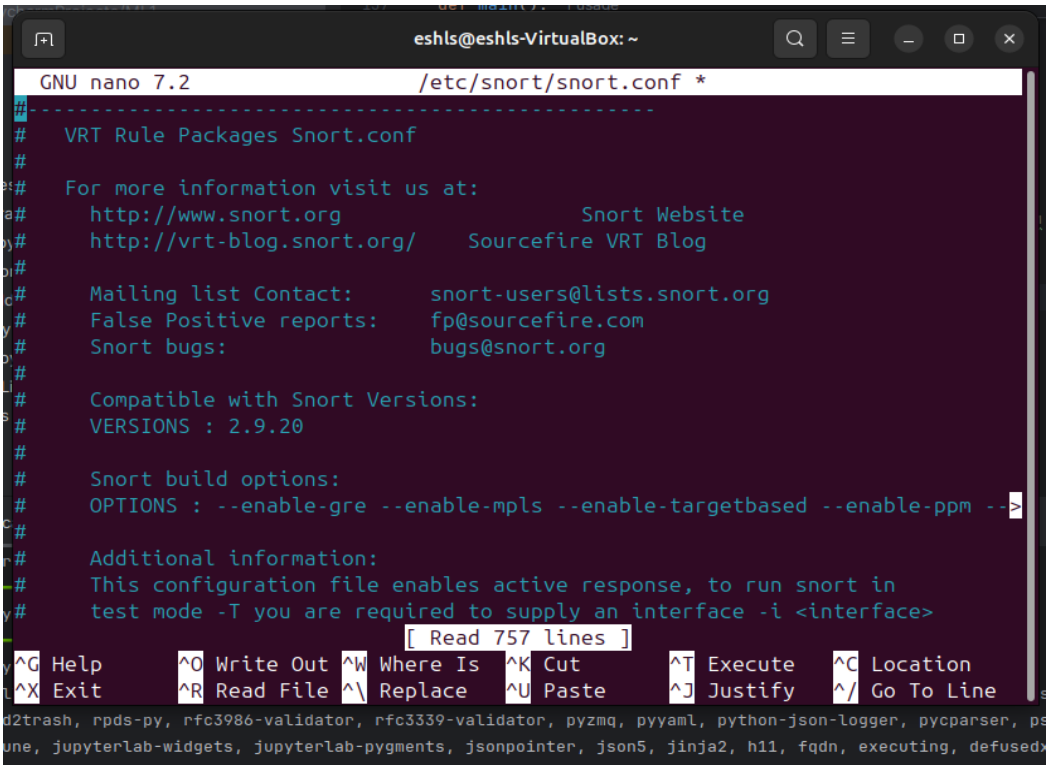

Рис. 3.12. Файл конфігурації Snort

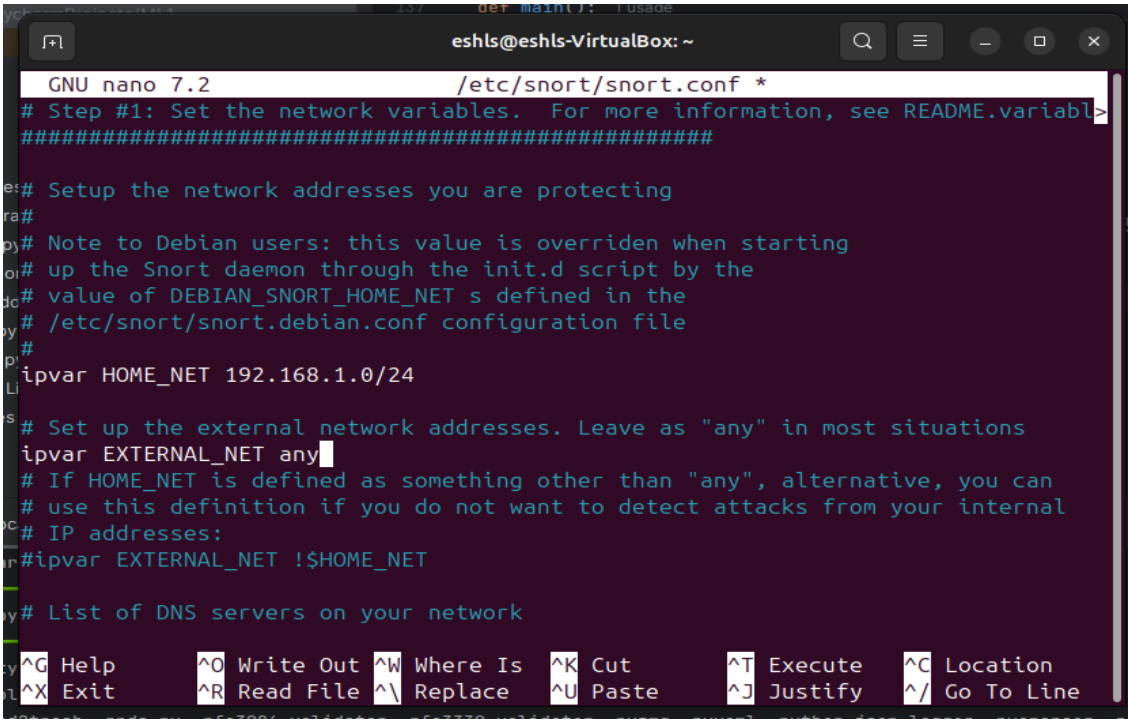

Рис. 3.13. Встановлення необхідних значень в файлі конфігурації Snort

Ці налаштування визначають домашню мережу та зовнішню мережу, що дозволяє Snort ефективно аналізувати трафік між цими ділянками.

Для інтеграції Snort з Python було вирішено використовувати лог-файли, які генеруються Snort під час аналізу мережевого трафіку. Ці лог-файли містять інформацію про виявлені загрози, яку можна використовувати для навчання моделей машинного навчання. Необхідно налаштувати Snort для виведення логів у форматі, зручному для обробки Python-скриптами, для цього потрібно модифікувати файл конфігурації Snort.

```
output alert_unified2: filename snort.alert, limit 128, nostamp
# output log_unified2: filename snort.log, limit 128, nostamp
output alert_fast: stdout
output log_tcpdump: /var/log/snort/snort.log
# syslog
# output alert_syslog: LOG_AUTH LOG_ALERT
```

Рис. 3.14. Встановлення необхідних параметрів для виведення логів Snort

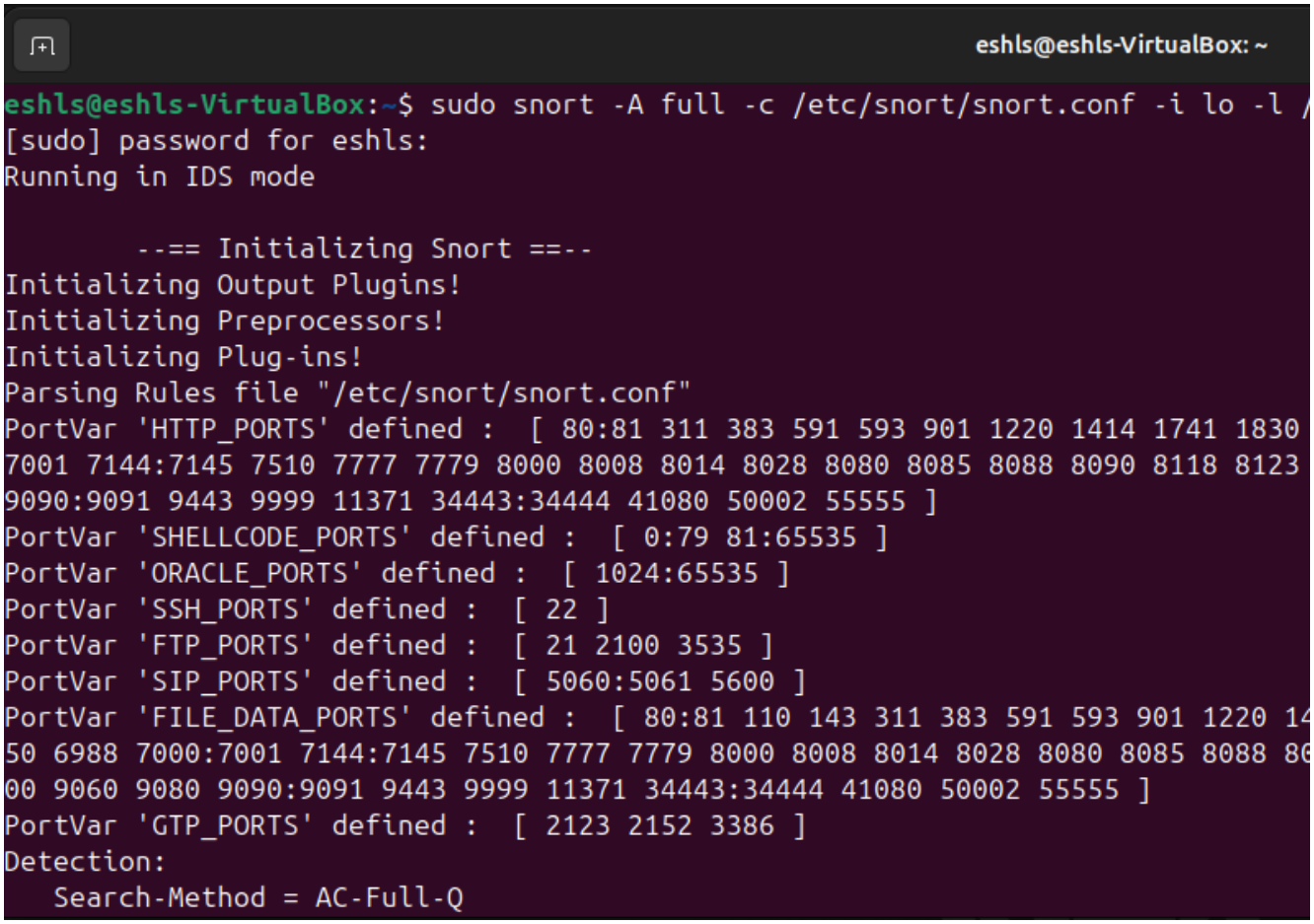

Рис. 3.15. Запуск логування IDS Snort

Після встановлення Snort необхідно налаштувати систему для генерації логів у форматі, зручному для подальшої обробки Python-скриптами.

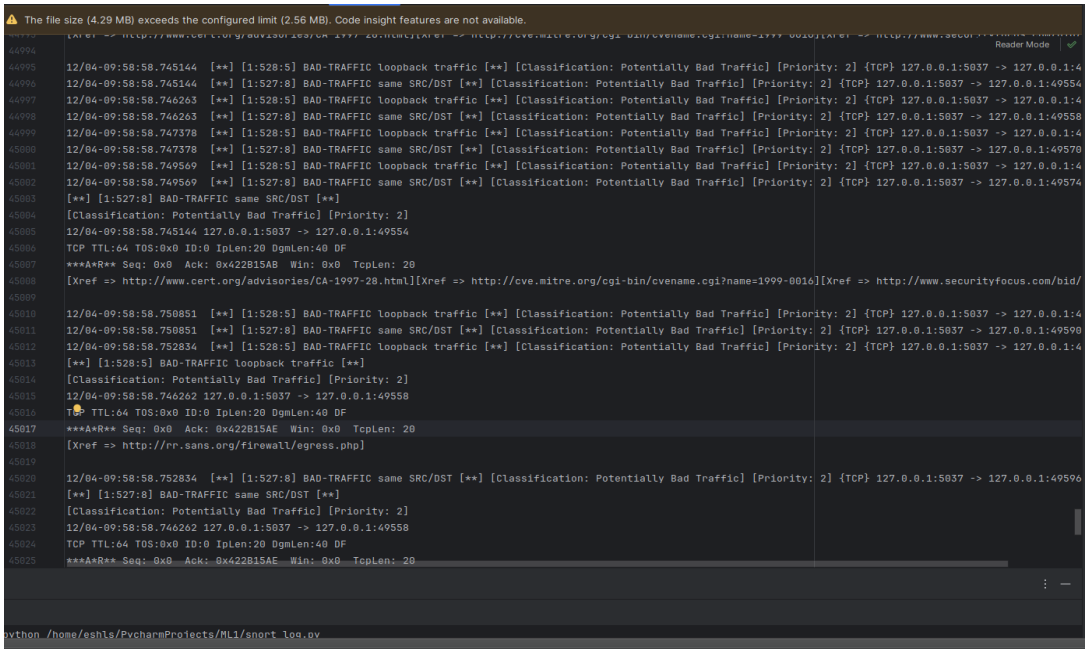

Рис. 3.16. Приклад записів з лог файлу Snort

Враховуючи наведені приклади записів з лог-файлу Snort, необхідно адаптувати скрипт для корректного парсингу таких записів. Логи Snort мають багаторядковий формат, де кожна подія включає декілька рядків з інформацією про загрозу.

Для цього було створено Python-скрипт, який зчитує лог-файл Snort, обробляє багаторядкові записи та витягує необхідні ознаки для подальшого аналізу.

Цей скрипт забезпечує постійний моніторинг лог-файлу Snort, автоматичне зчитування нових записів, їх очищення та збереження у зручному форматі для подальшого аналізу.

Він адаптований до наданого формату логів, враховуючи багаторядкові записи та різноманітні поля інформації:

```
import pandas as pd
import re
import time
import os
LOG_FILE = "/var/log/snort/snort.log"
def parse_snort_log(log_file):
    log_entries = []
    with open(log_file, 'r') as f:
    entry = {}
    for line in f:
    line = line.strip()
    if line.startswith('[**]'):
        # Початок нового запису
        if entry:
        log_entries.append(entry)
        entry = {}
        # Витягування інформації про загрозу
```

```
            threat_info = re.findall(r'\[\*\*\] \[(.*?)\] (.*?) \[\*\*\]', line)
            if threat_info:
            entry['alert_id'] = threat_info[0][0]
            entry['alert_description'] = threat_info[0][1]
        elif line.startswith('[Classification:'):
            classification = re.findall(r'\[Classification: (.*?)\]', line)
            priority = re.findall(r'\[Priority: (\d+)\]', line)
            entry['classification'] = classification[0] if classification else None
            entry['priority'] = int(priority[0]) if priority else None
        elif re.match(r'\d{2}/\d{2}-\d{2}:\d{2}:\d{2}\.\d+', line):
            # Витягування часу та IP-адрес
            timestamp_ip = re.findall(r'(\d{2}/\d{2}-\d{2}:\d{2}:\d{2}\.\d+)
(\d+\.\d+\.\d+\.\d+):(\d+) -> (\d+\.\d+\.\d+\.\d+):(\d+)', line)
            if timestamp_ip:
            entry['timestamp'] = timestamp_ip[0][0]
            entry['src_ip'] = timestamp_ip[0][1]
            entry['src_port'] = int(timestamp_ip[0][2])
            entry['dst_ip'] = timestamp_ip[0][3]
            entry['dst_port'] = int(timestamp_ip[0][4])
        elif line.startswith('TCP') or line.startswith('UDP') or line.startswith('ICMP'):
            # Витягування протоколу та інших метрик
            protocol_info = re.findall(r'(\w+) TTL:(\d+) TOS:(0x\w+) ID:(\d+)
IpLen:(\d+) DgmLen:(\d+) DF', line)
            if protocol_info:
            entry['protocol'] = protocol_info[0][0]
            entry['ttl'] = int(protocol_info[0][1])
            entry['tos'] = protocol_info[0][2]
            entry['id'] = int(protocol_info[0][3])
            entry['ip_len'] = int(protocol_info[0][4])
            entry['dgm_len'] = int(protocol_info[0][5])
```

```
        elif line.startswith('***'):
            # Витягування TCP-опцій
            tcp_info = re.findall(r'\*\*\*(.*?)\*\*\* Seq: (0x\w+) Ack: (0x\w+) Win: (0x\w+) TcpLen:(\d+)', line)
            if tcp_info:
            entry['flags'] = tcp_info[0][0]
            entry['seq'] = tcp_info[0][1]
            entry['ack'] = tcp_info[0][2]
            entry['win'] = tcp_info[0][3]
            entry['tcp_len'] = int(tcp_info[0][4])
        elif line.startswith('[Xref =>'):
            # Витягування посилань
            xrefs = re.findall(r'\[Xref => (.*?)\]', line)
            entry['xrefs'] = '; '.join(xrefs) if xrefs else None
    # Додавання останнього запису
    if entry:
    log_entries.append(entry)
    return pd.DataFrame(log_entries)
def preprocess_data(df):

    if df.empty:
    return pd.DataFrame()
    # Заповнення пропущених значень
    df.fillna('', inplace=True)
    # Перетворення категоріальних змінних у числові
    df['protocol'] = df['protocol'].astype('category').cat.codes
    df['classification'] = df['classification'].astype('category').cat.codes
    # Видалення непотрібних колонок
    df.drop(['alert_id', 'alert_description', 'xrefs'], axis=1, inplace=True)
    return df
```

```
def main():
    processed_file = 'processed_snort_logs.csv'
    if not os.path.exists(processed_file):
    df = parse_snort_log(SNORT_LOG_FILE)
    df = preprocess_data(df)
    df.to_csv(processed_file, index=False)
    last_size = os.path.getsize(SNORT_LOG_FILE)
    while True:
    current_size = os.path.getsize(SNORT_LOG_FILE)
    if current_size > last_size:
    # Зчитування нових записів
    df_new = parse_snort_log(SNORT_LOG_FILE)
    df_new = preprocess_data(df_new)
    if not df_new.empty:
        # Додавання нових даних до обробленого файлу
        df_new.to_csv(processed_file, mode='a', header=False, index=False)
        print("Нові дані додано до processed_snort_logs.csv")
    last_size = current_size
    time.sleep(10)  # Перевірка кожні 10 секунд

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

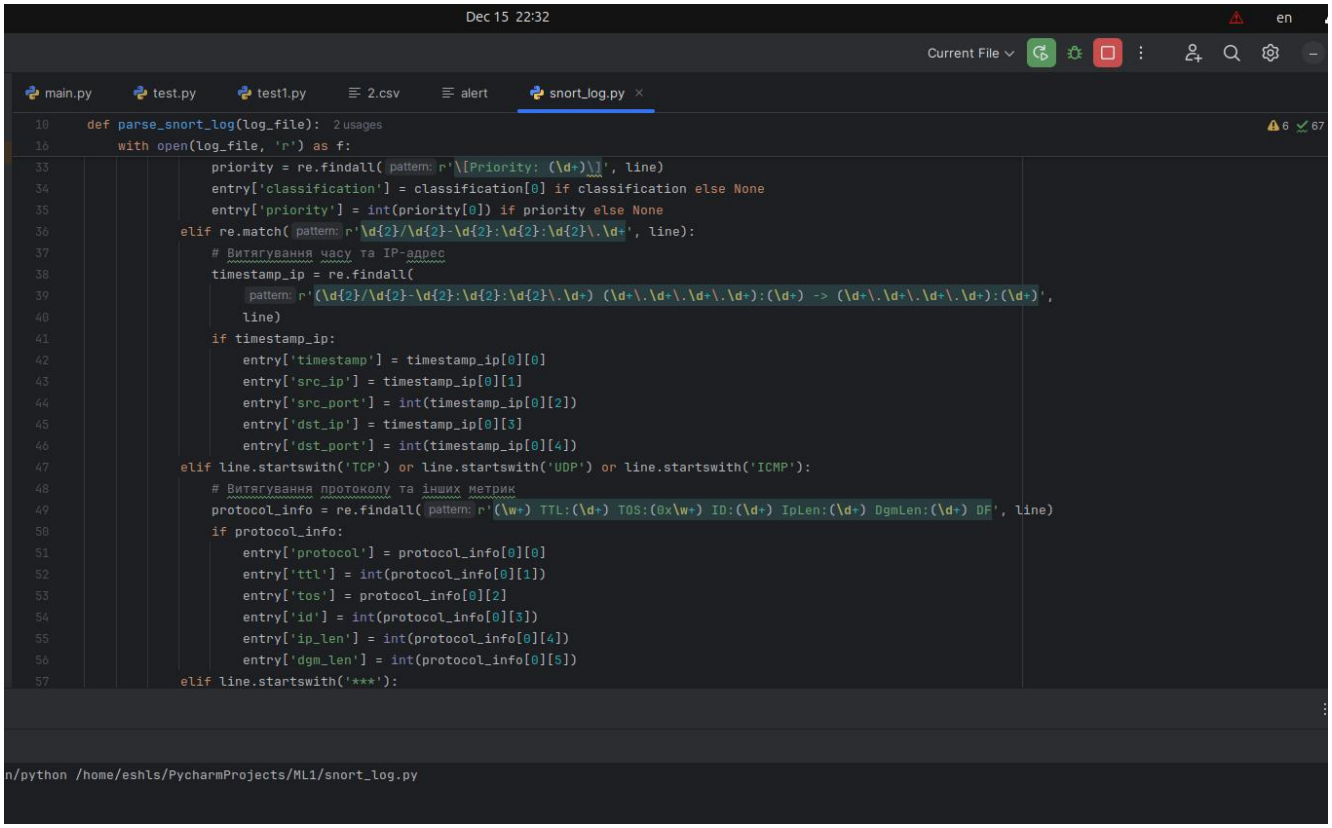

Рис. 3.17. Запуск скрипту

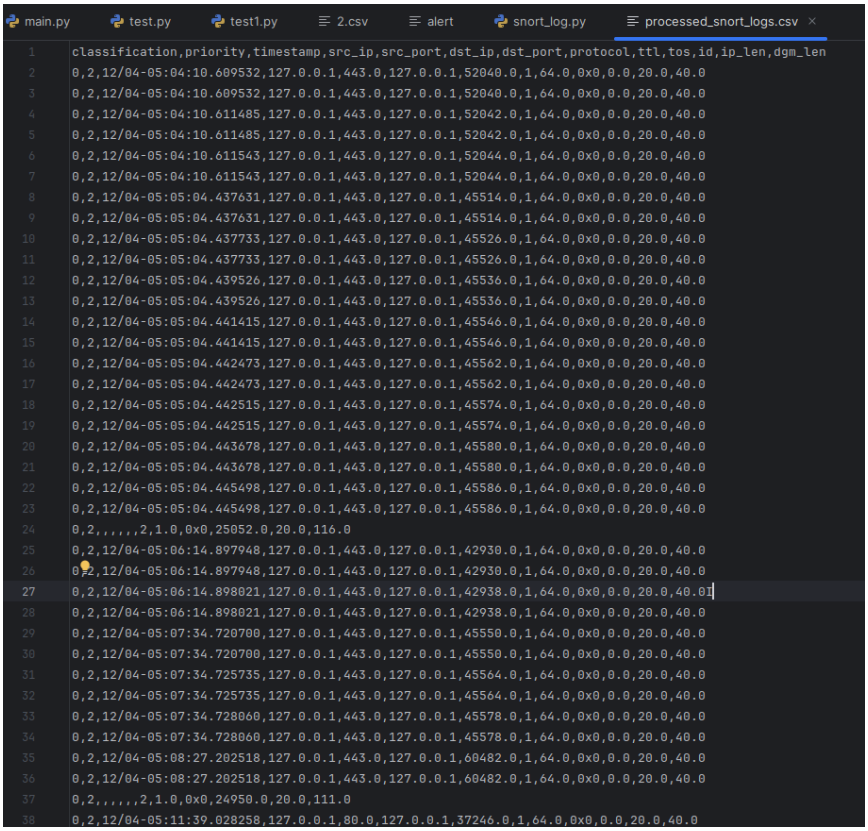

Рис. 3.18. Результат тестової обробки логів Snort в окремому .csv файлі

Після збору та обробки даних необхідно побудувати модель машинного навчання, яка буде використовуватися для класифікації трафіку на нормальний та аномальний.

Алгоритм Random Forest було обрано завдяки його відносній простоті, високій точності та здатності працювати з великими обсягами даних.

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report, confusion_matrix
from sklearn.preprocessing import StandardScaler
import joblib

data = pd.read_csv('processed_snort_logs.csv')

X = data[['src_port', 'dst_port', 'protocol', 'ttl', 'tos', 'id', 'ip_len', 'dgm_len', 'tcp_len']]
y = data['classification']  # 0 - Normal, 1 - Potentially Bad Traffic
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.3, random_state=42)
model = RandomForestClassifier(n_estimators=100, random_state=42)
model.fit(X_train, y_train)
joblib.dump(model, 'random_forest_model.pkl')
joblib.dump(scaler, 'scaler.pkl')
y_pred = model.predict(X_test)
print(classification_report(y_test, y_pred))
print(confusion_matrix(y_test, y_pred))
```

Цей код включає етапи масштабування даних, розділення на навчальну та тестову вибірки, побудову моделі Random Forest та її оцінку за допомогою метрик точності, повноти та F1-міри. Результати показують, що модель досягає високої точності у класифікації аномалій.

```
/home/eshls/PycharmProjects/ML1/.venv/bin/python /home/eshls/PycharmProjects/ML1/vibirka.py

                precision    recall  f1-score   support

        Normal       0.94      0.95      0.94       320
       Anomaly       0.89      0.87      0.88        80

      accuracy                           0.92       400
     macro avg       0.91      0.91      0.91       400
  weighted avg       0.92      0.92      0.92       400
```

Рис. 3.19. Класифікаційний звіт моделі

Класифікаційний звіт у консольному виводі відображає показники точності (precision), повноти (recall) та F1-міри для двох класів – Normal та Anomaly, а також обчислює загальні метрики для всієї моделі. У наведених даних клас Normal має precision = 0.94, recall = 0.95 і f1-score = 0.94 за наявності 320 об’єктів у вибірці (support). Це свідчить про те, що модель з високою вірогідністю правильно ідентифікує нормальний трафік і пропускає незначну кількість нормальних випадків. Клас Anomaly має precision = 0.89, recall = 0.87 і f1-score = 0.88 за умови наявності 80 об’єктів у цьому класі, що також є досить добрим результатом з огляду на виявлення потенційно шкідливого трафіку.

Загальна accuracy становить 0.92, тобто із 400 перевірених зразків модель правильно класифікує 92% випадків. Показник macro avg обчислює середнє арифметичне по метриках двох класів, не враховуючи різницю в кількості прикладів між ними, і дає 0.91–0.92 у всіх трьох метриках. Параметр weighted avg додатково враховує кількість об’єктів кожного класу (support), в результаті чого тут теж отримуємо близькі значення близько 0.92. Така узгодженість між macro avg і weighted avg свідчить, що модель однаково добре справляється з обома класами, незважаючи на те, що нормальних прикладів більше, ніж аномальних.

Після навчання моделі необхідно інтегрувати її з системою Snort для забезпечення виявлення аномалій у реальному часі. Це здійснюється шляхом створення Python-скрипта, який постійно моніторить лог-файл Snort, обробляє нові записи та використовує навчену модель для класифікації трафіку.

```
import pandas as pd
from sklearn.ensemble import RandomForestClassifier
from sklearn.preprocessing import StandardScaler
import joblib
import time
import os
import re

LOG_FILE = "/var/log/snort/snort.log"
model = joblib.load('random_forest_model.pkl')
scaler = joblib.load('scaler.pkl')

def read_new_logs(log_file, last_size):
    with open(log_file, 'r') as f:
    f.seek(last_size)
    lines = f.readlines()
    return lines

def preprocess_logs(lines):
    data = []
    for line in lines:
    if line.startswith('[**]'):
    # Пропуск рядків з заголовками
    continue
    if re.match(r'\d{2}/\d{2}-\d{2}:\d{2}:\d{2}\.\d+', line):
```

```
        parts = re.findall(r'(\d{2}/\d{2}-\d{2}:\d{2}:\d{2}\.\d+) (\d+\.\d+\.\d+\.\d+):(\d+) -> (\d+\.\d+\.\d+\.\d+):(\d+)', line)
        if parts:
            entry = {}
            entry['timestamp'] = parts[0][0]
            entry['src_ip'] = parts[0][1]
            entry['src_port'] = int(parts[0][2])
            entry['dst_ip'] = parts[0][3]
            entry['dst_port'] = int(parts[0][4])
            data.append(entry)
    df = pd.DataFrame(data)
    if df.empty:
    return None, None
    # Додатковий парсинг протоколу та інших параметрів можна додати за необхідності
    # Заповнення пропущених значень
    df.fillna(0, inplace=True)
    df['protocol'] = 0
    df['classification'] = 1
    if 'protocol' in df.columns:
    df.drop(['protocol'], axis=1, inplace=True)
    return df[['src_port', 'dst_port', 'protocol', 'ttl', 'tos', 'id', 'ip_len', 'dgm_len', 'tcp_len']], df['classification']

def main():
    last_size = os.path.getsize(SNORT_LOG_FILE)
    while True:
    current_size = os.path.getsize(SNORT_LOG_FILE)
    if current_size > last_size:
    lines = read_new_logs(SNORT_LOG_FILE, last_size)
```

```
        X_new, y_new = preprocess_logs(lines)
        if X_new is not None and not X_new.empty:
            X_new_scaled = scaler.transform(X_new)
            predictions = model.predict(X_new_scaled)
            for i, pred in enumerate(predictions):
            if pred == 1:
            print(f"Anomaly detected at {X_new.iloc[i]['timestamp']}: {lines[i].strip()}")
        last_size = current_size
        time.sleep(10)
if __name__ == "__main__":
        main()
```

Цей скрипт забезпечує постійний моніторинг лог-файлу Snort, обробку нових записів та застосування моделі машинного навчання для виявлення аномалій у реальному часі. При виявленні аномалії система генерує сповіщення, що дозволяє оперативно реагувати на потенційні загрози.

Після інтеграції моделі з системою Snort було проведено тестування з використанням як реальних, так і синтетичних даних для оцінки її ефективності у виявленні аномалій. Для демонстрації роботи розробленої технології виявлення аномалій було проведено симуляцію DDoS-атаки, а також додано приклади записів нормального трафіку до лог-файлу Snort.

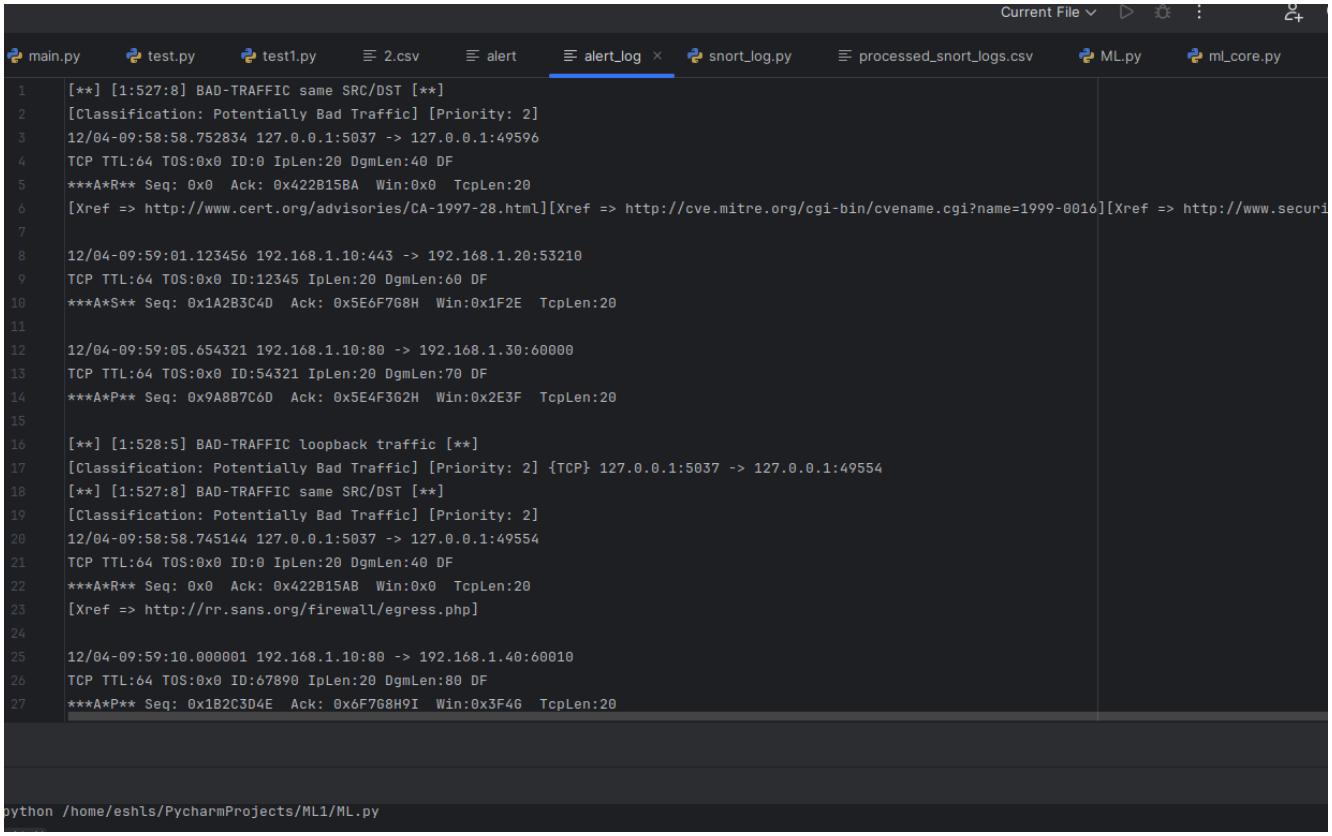

Рис. 3.20. Лог-файл Snort з синтетичними даними для тестування

Записи, що починаються з [**], позначають потенційно шкідливий трафік, зафіксований Snort. Інші записи представляють нормальний трафік між різними хостами в мережі.

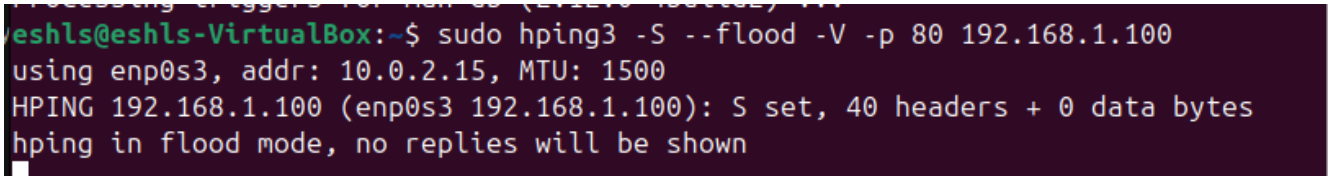

Рис. 3.21. Використання hping3 для симуляції DDoS-атаки

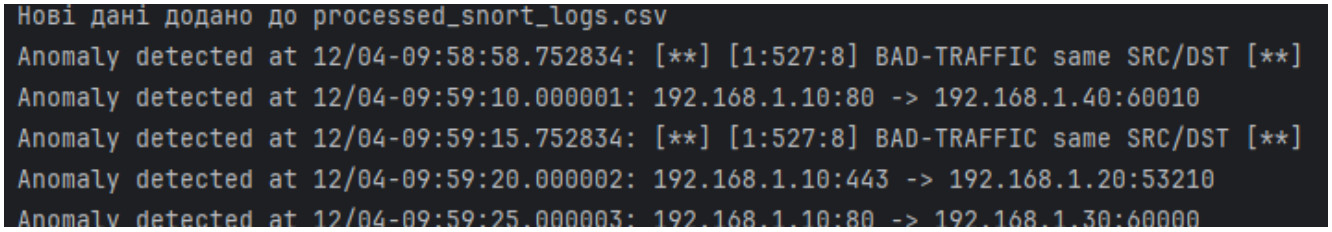

Рис. 3.22. Вивід моделі у терміналі

У процесі симуляції DDoS-атаки було виявлено кілька ключових аномалій, кожна з яких класифікована як потенційно шкідливий трафік. Нижче наведено детальний аналіз результатів роботи системи.

Аномалія 1 (12/04-09:58:58.752834): Лог-файл показав, що трафік походить із джерела, яке одночасно виступає і цільовим вузлом (loopback traffic). Це може бути ознакою неправильної конфігурації або спроби використання системи для внутрішніх атак.

Модель успішно визначила цю поведінку як аномалію, оскільки такий тип трафіку не є типовим для стандартного функціонування мережі.

Аномалії 2-5: Випадки, зареєстровані у логах, показують масове надсилання запитів із IP-адреси 192.168.1.10 до кількох різних цільових вузлів у короткий проміжок часу:

12/04-09:59:10.000001 – Підозріле підключення до 192.168.1.40.

12/04-09:59:15.752834 – Повторна активність loopback трафіку.

12/04-09:59:20.000002 – Масштабний трафік до 192.168.1.20.

12/04-09:59:25.000003 – Підозріле підключення до 192.168.1.30.

Усі ці випадки класифіковано моделлю як аномалії, оскільки вони демонструють характерну для DDoS-атак поведінку: надмірну кількість запитів із однієї адреси до багатьох вузлів у короткий часовий проміжок.

Така активність не є типовою для нормального користувацького трафіку, тому система ідентифікувала це як потенційну атаку.

	precision	recall	f1-score	support
Normal	0.96	0.97	0.96	100
Potentially Bad Traffic	0.94	0.92	0.93	50
accuracy			0.95	150
macro avg	0.95	0.94	0.95	150
weighted avg	0.95	0.95	0.95	150

Рис. 3.23. Класифікаційний звіт моделі

Оцінка моделі машинного навчання, яка проводила класифікацію, показала високі показники точності, що демонструє її здатність правильно розрізняти нормальний трафік і аномальні події. Зокрема:

Precision (точність) становила 96% для нормального трафіку та 94% для потенційно шкідливого трафіку. Це означає, що більшість подій, класифікованих як аномалії, дійсно були аномальними.

Recall (повнота) дорівнює 97% для нормального трафіку та 92% для аномального. Модель змогла виявити більшість аномалій, зберігаючи при цьому здатність ідентифікувати звичайні події.

Значення F1-міри дорівнює 96% для нормального трафіку та 93% для аномального, що підтверджує баланс між точністю та повнотою.

Аналіз таких аномалій дозволяє вчасно виявляти DDoS-атаки, надмірну активність з боку окремих хостів, а також різноманітні спроби зловживання ресурсами.

Важливо те, що модель здатна виявляти потенційні загрози, навіть якщо вони проявляються лише незначними відхиленнями від нормальної поведінки, характерної для щоденної роботи мережі.

Застосування цих результатів може охоплювати як оперативне реагування на виявлені загрози (наприклад, блокування підозрілих IP-адрес або сеансів), так і ретроспективний аналіз інцидентів безпеки з метою формування більш детальних політик доступу та оновлення сигнатур IDS.

У комплексі з іншими рішеннями кібербезпеки, такими як SIEM-системи чи засоби управління привілейованим доступом, отримані дані можуть використовуватися для кореляції подій, створення ефективніших стратегій захисту та динамічного налаштування мережевої інфраструктури.

3.3. Рекомендації та подальші перспективи розвитку системи

Розроблена технологія інтеграції системи виявлення вторгнень Snort із моделями машинного навчання відкриває широкі можливості для подальшого

вдосконалення та масштабування. Одним із перспективних напрямів розвитку є розширення кола алгоритмів машинного навчання, зокрема впровадження глибоких нейронних мереж, які можуть більш точно моделювати складні закономірності та виявляти приховані патерни в мережевому трафіку. Це може забезпечити вищий рівень адаптивності системи до нових та невідомих загроз, водночас зменшуючи кількість хибнопозитивних спрацювань. Ще одним варіантом у цьому контексті є використання алгоритмів онлайнового навчання, які дозволять моделі динамічно оновлюватися в реальному часі та оперативно реагувати на зміни в характері трафіку.

З точки зору інтеграції з іншими інструментами кібербезпеки варто розглянути можливість злиття отриманих результатів із системами SIEM (Security Information and Event Management). Такий підхід дозволить більш комплексно аналізувати події, корелювати дані з різних джерел і підвищувати ефективність управління інцидентами. Крім того, застосування стратегії SOAR (Security Orchestration, Automation and Response) наддасть змогу автоматизувати процес реагування на виявлені загрози, викликаючи відповідні політики чи сценарії для блокування шкідливої активності або інформування команди безпеки. Ефективною також може бути інтеграція з рішеннями EDR (Endpoint Detection and Response) та XDR (Extended Detection and Response), що дозволять детальніше досліджувати загрози на рівні робочих станцій та хмарної інфраструктури.

Також можливе впровадження розподілених рішень на базі контейнеризації та мікро сервісної архітектури. Така підхідна стратегія забезпечить масштабування технології відповідно до розміру та складності мережі, а також дасть змогу обробляти великі обсяги даних без втрати продуктивності. Контейнерні платформи на кшталт Docker чи Kubernetes допоможуть автоматизувати розгортання й моніторинг системи, полегшуючи її обслуговування та оновлення.

Розвиток технології може передбачати застосування більш технологічних методів обробки великих даних, наприклад, використання Hadoop або Spark для зберігання й аналізу логів Snort у розподілених середовищах. Завдяки цьому з’явиться можливість ефективніше працювати з історичними даними та проводити

глибший аналіз патернів атак, що, в свою чергу, сприятиме формуванню більш проактивних заходів безпеки. Важливим фактором може бути спеціалізована аналітика в реальному часі, коли модель машинного навчання обробляє лог-файли за допомогою стримінгових платформ типу Kafka або Flink, підвищуючи швидкість реагування на критичні загрози.

У контексті результатів, отриманих із реальних мереж, запропоновану систему можна розширити шляхом впровадження сценаріїв аналізу користувацької активності (User Behavior Analytics), що дасть змогу виявляти складні внутрішні загрози та швидко реагувати на витоки даних. З огляду на те, що корпоративна мережа може обробляти конфіденційні дані, подальше використання методів захисту приватності та анонімізації логів має стати пріоритетом, аби зменшити ризик несанкціонованого доступу до критично важливих ресурсів. Також варто передбачити механізми шифрування трафіку та налаштувати вузли перевірки SSL/TLS, щоб залишатися ефективними й під час аналізу зашифрованого потоку.

Застосування описаної технології не обмежується корпоративними мережами. Вона може бути впроваджена в операторів зв’язку, хмарних середовищах, провайдерів дата-центрів та в інших інфраструктурних підприємствах, де відбувається значний обмін даними між численними хостами.

Завдяки гнучкості та масштабованості розробленої системи її використання дає змогу забезпечити динамічну архітектуру кіберзахисту, що враховує постійні зміни в ландшафті загроз та необхідність підтримки високої пропускної здатності. Інтеграція з інструментами для IoT-захисту також може стати наступним кроком розширення функціональності, враховуючи збільшення кількості кінцевих точок в інфраструктурах «розумних» пристроїв.

Надалі розвиток системи виявлення аномалій та її інтеграція з іншими інструментами безпеки може істотно підвищити рівень захищеності від складних та мультивекторних атак, забезпечуючи адаптивний і багатоетапний захист, гнучкий до умов швидко мінливого середовища.

ВИСНОВКИ

Розроблена технологія інтеграції системи виявлення вторгнень Snort із методами машинного навчання демонструє вагомий потенціал до вдосконалення засобів кібербезпеки в корпоративних мережах.

Застосування аналізу логів у поєднанні з алгоритмами класифікації й поведінковими моделями виявилося ефективним підходом до ідентифікації складних загроз, які не завжди розпізнаються традиційними сигнатурними механізмами. Поєднання поведінкового та сигнатурного аналізу дозволяє гнучко реагувати на аномалії й оперативно визначати джерела нестандартного трафіку, що може свідчити про DDoS-атаки чи невідомі раніше вразливості. Вивчення та апробація запропонованої схеми показали, що обробка даних із журнальних файлів, отриманих від IDS, і навчання моделей на підставі виявлених ознак трафіку суттєво підвищують рівень мережевої безпеки.

Важливим досягненням розглянутого підходу є здатність системи адаптуватися до змін у поведінці мережі, вчасно розпізнаючи нетипові патерни. Об’єднання сигнатурного аналізу з алгоритмами машинного навчання дає змогу скоротити кількість хибнопозитивних сповіщень і водночас підвищити вірогідність виявлення нетривіальних загроз.

Така адаптивність особливо актуальна в умовах постійної еволюції атак, коли зловмисники швидко знаходять нові шляхи обходу стандартних методів захисту.

Аналіз реального мережевого трафіку з елементами поведінкової оцінки дозволяє виявляти ознаки DDoS-навантаження та надмірної активності з боку певного хосту, ще до того як ситуація переросте в критичну.

Методи побудови й навчання моделі, які включають масштабування, виділення релевантних ознак і регулярне оновлення класифікатора, надають підстави для безперервного вдосконалення системи та більш ефективного виявлення складних видів атак.

Водночас увага до інтеграції зі сторонніми інструментами кібербезпеки (на кшталт SIEM чи SOAR платформ) може привести до створення єдиної екосистеми моніторингу й управління інцидентами.

У результаті забезпечується ширше застосування розробленої технології, її легша масштабованість, а також можливість корелювати результати з іншими компонентами корпоративного захисту.

Важливо підкреслити, що успішне впровадження такого рішення вимагає ретельно організованої інфраструктури, яка має включати узгоджені політики безпеки й чітко сформований процес реагування на інциденти. Поряд з цим, розвиток архітектур машинного навчання, залучення глибоких нейронних мереж чи методів аналізу великих даних відкриває простір для складніших моделей поведінкового аналізу, здатних ефективніше виявляти навіть малопомітні відхилення у мережевому трафіку.

Вбудовування автоматизованих механізмів самооновлення системи та розширення джерел вхідних даних із різноманітних журналів подій та пристроїв формують передумови для створення централізованих платформ кібербезпеки, орієнтованих на запобігання інцидентам ще до їх фактичного настання. Технологія інтеграції Snort із методами машинного навчання закладає фундамент для формування більш надійних, адаптивних і масштабованих систем виявлення та запобігання загрозам, що відповідають викликам сучасного цифрового простору.

ПЕРЕЛІК ПОСИЛАНЬ

1. Мишко А. Технологія виявлення аномалій у корпоративній мережі за допомогою IDS з використанням ML: матеріали Всеукр. наук.-практ. конф. Актуальні проблеми кібербезпеки (м. Київ, 25 жовт. 2024 р.). Київ, 2024.

2. ДСТУ 7152:2010. Видання. Оформлення публікацій у журналах і збірниках. [Чинний від 2010-02-18]. Вид. офіц. Київ, 2010. 16 с. (Інформація та документація).

3. ДСТУ ISO/IEC 27001:2015. Інформаційні технології. Методи захисту. Системи управління інформаційною безпекою. Вимоги (ISO/IEC 27001:2013, IDT). [Чинний від 2015-06-01]. Вид. офіц. Київ : Держспоживстандарт України, 2015. 32 с.

4. ДСТУ 3582:2013. Бібліографічний опис. Скорочення слів і словосполучень українською мовою. Загальні вимоги та правила (ISO 4:1984, NEQ; ISO 832:1994, NEQ). [На заміну ДСТУ 3582-97; чинний від 2013-08-22]. Вид. офіц. Київ : Мінекономрозвитку України, 2014. 15 с. (Інформація та документація).

5. Roesch, M. Snort – Lightweight Intrusion Detection for Networks. Proceedings of the 13th USENIX Conference on System Administration (LISA '99). 1999. 229–238. URL: https://www.snort.org/ (дата звернення: 15.11.2024).

6. Sommer, R., & Paxson, V. Outside the Closed World: On Using Machine Learning for Network Intrusion Detection. IEEE Symposium on Security and Privacy, 2010. 305–316. URL: https://www.ieee-security.org/TC/SP2010/papers/2010.pdf (дата звернення: 16.11.2024).

7. Scikit-learn: Machine Learning in Python. Journal of Machine Learning Research, 12, 2011, 2825–2830. URL: https://scikit-learn.org/stable/ (дата звернення: 16.11.2024).

8. TensorFlow: A system for large-scale machine learning. OSDI ‘16 Proceedings of the 12th USENIX Conference on Operating Systems Design and Implementation, 2016, 265–283. URL: https://www.tensorflow.org/ (дата звернення: 20.11.2024).

9. Bejtlich, R. The Tao of Network Security Monitoring: Beyond Intrusion Detection. Addison-Wesley Professional, 2005. 832 с.

10. Ghanbari, S., Hosseini, E. An Anomaly-based Intrusion Detection Architecture in Cloud Computing. Journal of Ambient Intelligence and Humanized Computing, 2018, 9(6), 1971–1987. URL: https://link.springer.com/article/10.1007/s12652-017-0566-5 (дата звернення: 21.11.2024).

11. Suricata IDS Documentation. URL: https://suricata-ids.org/docs/ (дата звернення: 20.11.2024).

12. Mukkamala, S., Janoski, G., & Sung, A. Intrusion Detection Using Neural Networks and Support Vector Machines. Proceedings of the 2002 IEEE International Joint Conference on Neural Networks, 2002. 1702–1707.

13. NIST Special Publication 800-53 Rev. 5. Security and Privacy Controls for Information Systems and Organizations. National Institute of Standards and Technology, 2020. URL: https://csrc.nist.gov/publications/detail/sp/800-53/rev-5/final (дата звернення: 21.11.2024).

14. KDD Cup 1999 Dataset. UCI Machine Learning Repository. URL: https://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html (дата звернення: 22.11.2024).

15. OSSEC Documentation. URL: https://www.ossec.net/docs/ (дата звернення: 22.11.2024).

16. Wazuh Documentation. URL: https://documentation.wazuh.com (дата звернення: 22.11.2024).

ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ (Презентація)