Міністерство освіти і науки України
Харківський національний університет імені В. Н. Каразіна
Навчально-науковий інститут комп'ютерних наук та штучного інтелекту

Кафедра кібербезпеки інформаційних систем, мереж і технологій

До захисту допущено
Кафедрою КІСМіТ протокол №____ від «___» грудня 2025 р.

завідувач кафедри _____ Марина ЄСІНА
(підпис) *(ім'я, прізвище)*
«___» грудня 2025 р.

Кваліфікаційна робота

здобувача другого (магістерського) рівня вищої освіти

Дослідження застосування алгоритмів машинного навчання для виявлення вторгнень у мережі

(назва роботи)

Спеціальність (спеціалізація) 125 «Кібербезпека та захист інформації»

Освітня програма «Безпека інформаційних і комунікаційних систем»

Виконавець _____ Максим БЛІНОВ
(підпис) *(ім'я, прізвище)*

Науковий керівник _____ Ігор СВАТОВСЬКИЙ
(підпис) *(ім'я, прізвище)*

Харків – 2025

РЕФЕРАТ

Пояснювальна записка до дипломної роботи складається зі вступу, трьох розділів, висновків, списку використаних джерел і містить 64 сторінки, 18 рисунків, 5 таблиць, 1 додатоку, 26 найменувань використаних джерел.

Метою роботи є дослідження ефективності систем виявлення вторгнень на основі сигнатурного аналізу (IDS Suricata) та оцінка її ефективності після інтеграції моделі машинного навчання.

Об’єкт дослідження – мережеві системи, інфраструктури моніторингу трафіку та засоби виявлення вторгнень.

Предмет дослідження – методи сигнатурного та інтелектуального аналізу мережевого трафіку для виявлення атак.

У роботі як основні методи дослідження використано аналіз мережевого трафіку, тестування атак у лабораторному середовищі, алгоритми машинного навчання для класифікації подій, а також методи оцінки ефективності систем виявлення вторгнень. В основу дослідження були покладені сучасні підходи до кіберзахисту відповідно до рекомендацій NIST.

У результаті роботи було досліджено ефективність базової IDS Suricata під час різних типів атак, а також проведено навчання та інтеграцію моделі машинного навчання для покращення точності класифікації подій. Під час тестування було отримано значне скорочення кількості дублюючих та неінформативних сповіщень, підвищення точності й стабільності виявлення атак та зменшення інформаційного шуму. Комбінована система IDS+ML продемонструвала здатність ефективніше виявляти сканування, brute-force атаки та аномальні HTTP-запити, що сприяє підвищенню загального рівня безпеки мереж.

Результати дослідження можуть бути використані для вдосконалення існуючих систем кіберзахисту, оптимізації обробки алертів.

Ключові слова: IDS, SURICATA, МАШИННЕ НАВЧАННЯ, ВИЯВЛЕННЯ АТАК, МЕРЕЖЕВИЙ ТРАФІК, АНОМАЛІЇ, КІБЕРБЕЗПЕКА.

ABSTRACT

The explanatory note to the thesis consists of an introduction, three chapters, conclusions, a list of references, and includes 64 pages, 18 figures, 5 tables, 1 appendix, and 26 referenced sources.

The aim of this work is to investigate the effectiveness of an intrusion detection system based on signature analysis (Suricata IDS) and to evaluate its performance after integrating a machine learning model. The object of the study is network systems, traffic monitoring infrastructures, and intrusion detection mechanisms.

The subject of the study is signature-based and intelligent methods of network traffic analysis for detecting cyberattacks.

The research employs methods of network traffic analysis, controlled attack testing in a laboratory environment, machine learning algorithms for event classification, and techniques for evaluating the efficiency of intrusion detection systems. The study is based on modern cybersecurity approaches in accordance with NIST recommendations.

As a result of the work, the effectiveness of the baseline IDS Suricata under various types of attacks was evaluated, and a machine learning model was trained and integrated to improve the accuracy of event classification. Experimental testing showed a significant reduction in duplicate and non-informative alerts, an increase in detection accuracy and stability, and a substantial decrease in information noise. The combined IDS+ML system demonstrated an enhanced ability to detect port scans, brute-force attempts, and anomalous HTTP-requests, contributing to a higher overall level of network security.

The results of the research can be used to improve existing cybersecurity systems and to optimize alert processing.

Keywords: IDS, SURICATA, MACHINE LEARNING, ATTACK DETECTION, NETWORK TRAFFIC, ANOMALIES, CYBERSECURITY.

ЗМІСТ

ПЕРЕЛІК ПОЗНАЧЕНЬ І СКОРОЧЕНЬ……..4
ВСТУП ……5
1 АНАЛІЗ ЗАГРОЗ ТА АТАК НА МЕРЕЖІ ТА МЕТОДИ ПРОТИДІЇ
1.1. Огляд основних видів атак на мережі.....7
1.1.1. Актуальність проблеми безпеки мереж.....7
1.1.2. Класифікація видів атак на мережі.....8
1.1.3. Аналіз інструментів для проведення атак та їх виявлення.....9
1.2. Огляд існуючих методів та засобів виявлення вторгнень.....11
2 АНАЛІТИЧНИЙ ОГЛЯД ЗАСТОСУВАННЯ МАШИННОГО НАВЧАННЯ ДЛЯ ВИЯВЛЕННЯ ВТОРГНЕНЬ У МЕРЕЖІ
2.1. Огляд відомих алгоритмів машинного навчання та їх застосування....13
2.2. Застосування машинного навчання у сфері кібербезпеки.....15
2.3. Методи використання нейронних мереж для аналізу атак.....17
2.4 Використання методу Random Forest для аналізу логів.....19
3 МОДЕЛЮВАННЯ ТА АНАЛІЗ СИСТЕМИ ВИЯВЛЕННЯ ВТОРГНЕНЬ
3.1. Побудова та налаштування мережі для тестування моделі.....28
3.2. Вибір та підготовка даних для навчання моделей.....32
3.3. Процес навчання моделей машинного навчання 40
3.4. Тестування та оцінка ефективності системи.....45
3.4.1. Оцінювання ефективності IDS Suricata.....45
3.4.2. Оцінювання ефективності IDS системи на основі машинного навчання.....51
3.4.3. Порівняння результатів ефективності.....57
ВИСНОВКИ.....62
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ.....65
ДОДАТОК А.....68

ПЕРЕЛІК ПОЗНАЧЕНЬ І СКОРОЧЕНЬ

IDS – Intrusion Detection System (система виявлення вторгнень)

IPS – Intrusion Prevention System (система запобігання вторгненням)

ML – Machine Learning (машинне навчання)

AI – Artificial Intelligence (штучний інтелект)

NIDS – Network Intrusion Detection System (мережева система виявлення вторгнень)

HIDS – Host Intrusion Detection System (хостова система виявлення вторгнень)

SIEM – Security Information and Event Management (керування подіями та інформацією безпеки)

NIST – National Institute of Standards and Technology (Національний інститут стандартів і технологій)

ISO – International Organization for Standardization (Міжнародна організація зі стандартизації)

TCP – Transmission Control Protocol

UDP – User Datagram Protocol

HTTP – HyperText Transfer Protocol

SQLi – SQL Injection (ін’єкція в SQL-запит)

DoS – Denial of Service (відмова в обслуговуванні)

DDoS – Distributed Denial of Service (розподілена відмова в обслуговуванні)

PCAP – Packet Capture (файл або формат перехопленого трафіку)

JSON – JavaScript Object Notation

API – Application Programming Interface

KNN – k-Nearest Neighbors (метод k-ближчих сусідів)

RF – Random Forest (випадковий ліс)

DT – Decision Tree (дерево рішень)

CSV – Comma-Separated Values (табличний формат даних)

VM – Virtual Machine (віртуальна машина)

ВСТУП

Актуальність роботи. У сучасних комп'ютерних мережах рівень загроз інформаційній безпеці невпинно зростає, а методи атак стають дедалі складнішими, динамічнішими та важче передбачуваними. З поширенням інтегрованих корпоративних інфраструктур, хмарних сервісів і високошвидкісних каналів зв'язку зловмисники застосовують сканування мереж, автоматизовані засоби підбору паролів, складні атаки прикладного рівня, а також різні види DDoS-та мережевих flood-атак [1]. Традиційні сигнатурні системи виявлення вторгнень (IDS), попри їхню поширеність та ефективність у типових сценаріях, часто стикаються з обмеженнями — зокрема, надмірним обсягом дублюючих сповіщень, недостатньою здатністю до виявлення нових або модифікованих атак, а також високим рівнем інформаційного шуму [1, 14, 21]. У цих умовах особливої актуальності набувають інтелектуальні підходи, що дозволяють автоматично аналізувати трафік, виявляти приховані закономірності та забезпечувати більш точне виявлення некоректних або шкідливих дій. Застосування алгоритмів машинного навчання у сфері виявлення вторгнень відкриває нові можливості для створення адаптивних і високоточних систем [2, 15, 16]. На відміну від класичних IDS, моделі машинного навчання здатні аналізувати поведінкові характеристики трафіку, виявляти аномалії та зменшувати кількість хибнопозитивних спрацьовувань, що є критично важливим для інформаційної безпеки [3]. Інтеграція штучного інтелекту у такі системи забезпечує глибшу оцінку подій, точнішу класифікацію атак та підвищує ефективність роботи аналітиків або автоматизованих засобів реагування.

Метою роботи є дослідження ефективності систем виявлення вторгнень на основі сигнатурного аналізу (IDS Suricata) та оцінка її ефективності після інтеграції моделі машинного навчання.

Об'єкт дослідження – мережеві системи, інфраструктури моніторингу трафіку та засоби виявлення вторгнень.

Предмет дослідження – методи сигнатурного та інтелектуального аналізу подій, а також алгоритми машинного навчання, що застосовуються для класифікації мережевих атак.

Методи дослідження. У роботі використовувалися методи аналізу мережевого трафіку, моделювання атак у тестовому середовищі, алгоритми машинного навчання для класифікації подій, статистичні методи оцінювання ефективності моделей.

Практичне значення одержаних результатів. У рамках роботи було проведено порівняння двох моделей роботи IDS: традиційної Suricata без модифікацій та гібридної системи, доповненої моделлю машинного навчання. Отримані результати дозволили визначити ступінь впливу ML-алгоритмів на точність виявлення атак, рівень хибнопозитивних сповіщень та загальну інформативність логів. Виявлено, що застосування інтелектуальних методів істотно підвищує якість аналізу, робить систему більш стійкою до інформаційного шуму та зменшує навантаження на фахівців із кібербезпеки. Результати можуть бути використані для вдосконалення існуючих корпоративних систем моніторингу, розробки нових підходів до фільтрації трафіку та впровадження більш ефективних гібридних IDS-рішень у мережах.

1 АНАЛІЗ ЗАГРОЗ ТА АТАК НА МЕРЕЖІ ТА МЕТОДИ ПРОТИДІЇ

1.1 Огляд основних видів атак на мережі

1.1.1 Актуальність проблеми безпеки мереж

Сучасні комп’ютерні мережі є фундаментальною основою для функціонування інформаційної інфраструктури більшості державних, комерційних та приватних організацій. У зв’язку з масштабною цифровізацією, розвитком електронних сервісів, поширенням хмарних технологій і збільшенням кількості пристроїв, підключених до мережі, питання безпеки стає особливо важливим [4 - 6]. Кожного року обсяги мережевого трафіку зростають, з’являються нові протоколи, нові програмні засоби і нові моделі взаємодії, а разом із ними — нові вразливості та потенційні вектори атак. Водночас, світовий рівень кіберзлочинності невпинно зростає, оскільки атаки стають дедалі доступнішими через появу автоматизованих інструментів, платформ для обміну шкідливим кодом, а також «кіберзлочинності як сервісу», що дозволяє навіть непідготовленим користувачам проводити складні атаки з мінімальними витратами.

Проблема посилюється зростанням кількості IoT-пристроїв, які часто мають низький рівень безпеки та стають легкими мішенями для бот-мереж. Масштабні DDoS-атаки, засновані на використанні тисяч або мільйонів заражених пристроїв, стали звичним явищем, і їхня середня інтенсивність з кожним роком збільшується [5]. Нові формати атак, включно зі складними багатоступеневими атаками, атакою на ланцюг постачання, використанням штучного інтелекту для автоматизації вибору цілей, роблять класичні системи захисту недостатньо ефективними. У цих умовах актуальність проблеми безпеки мереж полягає не лише в необхідності протидії відомим загрозам, але й у здатності адаптуватися до нових, ще не досліджених моделей атак.

Негативні наслідки порушення безпеки мереж можуть включати витік конфіденційної інформації, порушення роботи критичних сервісів, фінансові втрати, підрив репутації організації, а у випадку державного сектору —

загрози національній безпеці. Новітні покоління атак, які поєднують соціальну інженерію, технічні експлойти та приховане проникнення в мережу, змушують фахівців постійно вдосконалювати системи моніторингу та виявлення вторгнень [6]. Усе це робить проблему безпеки мереж не лише актуальною, але й однією з найважливіших у сфері інформаційних технологій.

1.1.2 Класифікація видів атак на мережі

Атаки на мережі можна класифікувати за різними ознаками. Однак загалом усі вони спрямовані на порушення трьох ключових властивостей інформації: конфіденційності, цілісності та доступності. Порушення конфіденційності передбачає несанкціоноване отримання доступу до інформації через пасивне прослуховування трафіку або активне втручання у комунікацію [7]. Наприклад, перехоплення незашифрованого трафіку у локальній мережі дає змогу атакувальнику отримати доступ до облікових даних, електронної пошти, службових повідомлень або внутрішніх API [7]. Особливо небезпечні атаки типу «людина посередині», що дозволяють не лише перехоплювати, а й модифікувати трафік між учасниками комунікації, залишаючись при цьому непоміченими.

Порушення доступності зазвичай реалізується через атаки типу DoS або DDoS, що полягають у створенні надмірного навантаження на сервер або мережевий канал, унаслідок чого легітимні користувачі не можуть отримати доступ до ресурсів. Сучасні DDoS-атаки використовують бот-мережі, які складаються з великої кількості заражених пристроїв — серверів, роутерів, домашніх камер або промислового обладнання. Використання протоколів amplification, як-от DNS або NTP, дозволяє зловмисникам значно збільшувати силу атаки, не витрачаючи власних ресурсів [8].

Порушення цілісності даних передбачає модифікацію інформації під час передавання або зберігання. Наприклад, атакувальник може змінити вміст мережевого пакета, підробити відповіді DNS, модифікувати конфігураційні файли або впровадити фальшиві записи, що згодом можуть бути використані

для подальшої ескалації прав. Такі атаки часто здійснюються після того, як зловмисник отримує певний рівень доступу до мережевої інфраструктури.

Особливу групу складають атаки на протоколи. Наприклад, недоліки у реалізації TCP можна використовувати для відправки пакетів скидання з'єднання або для проведення сканування закритих портів. Вразливості у протоколах маршрутизації, зокрема BGP, можуть призвести до перенаправлення трафіку через недовірені вузли, що відкриває можливості для масштабного шпигунства або модифікації трафіку. У локальних мережах популярними є атаки на ARP, які дозволяють перенаправити трафік через машину зловмисника шляхом підміни MAC-адрес [8].

Атаки на доступ зазвичай включають brute-force облікових даних, підбір паролів, експлуатацію вразливостей операційних систем або сервісів, використання шкідливого ПЗ і бекдорів. Після проникнення в систему зловмисник може виконати ескалацію привілеїв, отримати доступ до інших сегментів мережі, встановити постійні механізми контролю або викрасти важливі дані.

1.1.3 Аналіз інструментів для проведення атак та їх виявлення

Світ інформаційної безпеки характеризується великою кількістю інструментів, які можуть бути використані як для тестування безпеки систем, так і для проведення реальних атак. Їх функціонал охоплює всі етапи атаки, включаючи збір інформації, сканування, експлуатацію вразливостей, постексплуатацію та приховування слідів. Серед найпопулярніших інструментів для сканування мереж є Nmap, який дозволяє виконувати детальне сканування портів, виявляти активні хости, визначати версії сервісів і операційних систем. Masscan виконує подібні функції, але із суттєвою перевагою щодо швидкості, що робить його ефективним засобом для аналізу великих діапазонів IP-адрес [9].

Одним із найпотужніших інструментів у сфері тестування безпеки є Metasploit Framework. Ця платформа містить тисячі готових експлойтів, shell-

кодів, модулів для виконання атак, а також інструменти для післяексплуатаційних дій. Завдяки автоматизації процесів зловмисники можуть проводити атаки без глибокого розуміння принципів роботи мережевих протоколів. Інструменти на кшталт Wireshark і tcpdump дозволяють проводити аналіз мережевого трафіку на низькому рівні, що необхідно для виявлення аномалій, реконструкції сесій або дослідження шкідливих пакетів. Утиліти типу hping3 дають можливість надсилати спеціально сформовані пакети, імітувати різні мережеві взаємодії, тестувати брандмауери та IDS [9].

Ефективні атаки на локальні мережі часто проводяться за допомогою Bettercap або Ettercap, які дозволяють виконувати ARP-підміну, перехоплювати HTTP- і HTTPS-трафік, аналізувати структуру мережі та впроваджувати шкідливі пакети. Застосування цих інструментів робить можливими атаки типу MITM, а також ін'єкції скриптів або модифікацію HTTP-вмісту у реальному часі.

Для brute-force атак часто використовують Hydra, Medusa або Ncrack, які дозволяють атакувати широкий спектр мережевих протоколів, що використовують парольну автентифікацію. Це дає змогу зловмисникам отримувати доступ до SSH, RDP, Telnet, FTP або SMTP, якщо облікові дані недостатньо захищені [9].

Протидія цим атакам неможлива без відповідних засобів моніторингу, аналізу та виявлення вторгнень. Найпоширенішими засобами є системи IDS/IPS, зокрема Suricata, Snort та Zeek. Suricata підтримує високорівный аналіз протоколів, багатопотокову обробку, сигнатури, поведінковий аналіз, а також генерує структуровані журнали подій у форматі JSON (eve.json), які широко використовуються для машинного аналізу. Snort є класичною сигнатурною системою, яка давно зарекомендувала себе як ефективний засіб для середніх і малих мереж. Zeek (раніше — Bro) орієнтований на поведінковий аналіз і створення детальних логів, які можна корелювати для складних атак. SIEM-системи, які об'єднують логи з різних джерел,

дозволяють виявляти багатоступеневі атаки, аналізувати інциденти та формувати автоматичні реакції на них [10].

1.2 Огляд існуючих методів та засобів виявлення вторгнень

Системи виявлення вторгнень є невід'ємною частиною сучасної мережевої безпеки. Їх основною функцією є аналіз трафіку або подій на хості з метою виявлення потенційно шкідливої активності. Методи виявлення можна умовно поділити на сигнатурні, аномалійні, поведінкові та гібридні. Сигнатурний підхід передбачає використання бази даних відомих сигнатур шкідливих атак. Порівнюючи трафік із цими сигнатурами, система може швидко визначити відомі загрози. Однак слабкою стороною є нездатність виявляти нові, невідомі атаки або модифікації старих експлойтів [10]. У таких випадках сигнатурна база потребує постійного оновлення, і навіть тоді залишається ризик, що нові атаки залишаться непоміченими.

Аномалійні методи базуються на побудові моделі нормальної поведінки мережі. Це може бути статистична модель, поведінкова модель або модель, основана на машинному навчанні. Після створення такої моделі система аналізує поточні події та визначає, наскільки вони відповідають нормі. Якщо подія виходить за межі встановлених параметрів, система позначає її як потенційно шкідливу. Основною перевагою є здатність виявляти невідомі атаки. Проте недоліком залишається висока кількість хибнопозитивних результатів, які можуть перевантажити аналітиків безпеки [8].

Поведінковий підхід дозволяє аналізувати взаємодію користувачів, хостів, протоколів і сервісів. Це означає, що система може виявляти відхилення в поведінці користувачів або сервісів, що часто є індикатором внутрішніх загроз, компрометації облікових записів або зловживань. Гібридні методи об'єднують сигнатурні та аномалійні підходи, що дозволяє досягти високої точності і водночас зменшити кількість хибних спрацювань [10].

Сучасні системи виявлення вторгнень усе частіше використовують машинне навчання. Це зумовлено тим, що кількість логів, які генерує IDS,

може досягати сотень гігабайт за короткий час. Ручний аналіз таких обсягів неможливий, тому штучний інтелект стає допоміжним, а іноді й основним інструментом для обробки даних. Моделі машинного навчання можуть автоматично класифікувати події з файлу eve.json, визначати типи атак, відфільтровувати хибні спрацювання та формувати більш точні звіти. Використання алгоритмів Random Forest, SVM, нейронних мереж, дерев рішень і градієнтного бустингу суттєво підвищує ефективність роботи IDS, дозволяючи швидше реагувати на загрози і зменшувати навантаження на фахівців [9].

2 АНАЛІТИЧНИЙ ОГЛЯД ЗАСТОСУВАННЯ МАШИННОГО НАВЧАННЯ ДЛЯ ВИЯВЛЕННЯ ВТОРГНЕНЬ У БЕЗДРОТОВІ МЕРЕЖІ

2.1 Огляд існуючих алгоритмів машинного навчання та їх застосування

З розвитком цифрових технологій та зростанням обсягів даних, що циркулюють у комп'ютерних мережах, машинне навчання стало одним із ключових інструментів для автоматизації процесів виявлення вторгнень та кіберзагроз. Алгоритми машинного навчання дозволяють системам адаптуватися до поведінки мережевого трафіку, виявляти приховані закономірності та реагувати на нові типи атак, які раніше не були відомі. На відміну від традиційних методів, заснованих на сигнатурах, підходи машинного навчання здатні аналізувати багатовимірні простори ознак і виявляти складні аномалії, характерні для шкідливої активності [9]. Тому в сучасних IDS вони поступово стають основою модулів поведінкового аналізу.

Одним із найбільш використовуваних алгоритмів у системах виявлення вторгнень є методи класифікації, що дозволяють визначати, чи належить певний тип трафіку до категорії легітимного чи шкідливого. Серед найбільш поширених алгоритмів класифікації виділяють дерева рішень, методи опорних векторів, логістичну регресію, байєсівські класифікатори, k-Nearest Neighbors, ансамблеві методи та нейронні мережі різних архітектур. Кожен із цих алгоритмів має свої переваги та недоліки, які визначаються обсягом навчальних даних, структурою ознак, лінійністю або нелінійністю розподілу мережевої активності та вимогами до обчислювальних ресурсів.

Дерева рішень часто використовуються у задачах мережевої безпеки через свою інтерпретованість та швидкість роботи. Вони здатні швидко будувати правила для класифікації потоків, але можуть бути схильні до перенавчання. Для усунення цих недоліків застосовуються ансамблеві алгоритми, такі як Random Forest та Gradient Boosting. Random Forest дозволяє формувати велику кількість дерев і голосувати за результат класифікації, забезпечуючи високу точність і стійкість до шуму в даних. Gradient Boosting,

натомість, навчає дерева послідовно, кожного разу компенсуючи помилки попередніх моделей, що дає можливість досягати дуже високої точності навіть у складних задачах багатокласової класифікації атак [11-9].

Метод опорних векторів також широко застосовується у задачах виявлення вторгнень завдяки здатності працювати з високовимірними просторами ознак та забезпечувати високий рівень узагальнення. Його використання стає особливо ефективним у задачах, де існує складна межа між різними класами трафіку. Проте метод SVM має складність навчання, яка погано масштабується на дуже великі набори мережевих даних.

Алгоритми кластеризації застосовуються для виявлення аномалій без необхідності наявності мічених даних. Найпопулярнішими серед них є k-means, DBSCAN та методи щільнісної кластеризації [11]. Ці методи дозволяють будувати моделі поведінки нормального трафіку та визначати відхилення, які можуть свідчити про вторгнення. Вони особливо корисні у випадках, коли потрібно виявляти нові, невідомі раніше типи атак, які не містяться у навчальних вибірках для алгоритмів класифікації.

Крім того, важливою частиною застосування машинного навчання у сфері кібербезпеки є методи зменшення розмірності, такі як Principal Component Analysis та t-SNE. Вони дозволяють виділяти ключові ознаки, зменшувати вплив шуму та полегшувати роботу класифікаторів і алгоритмів аномалійного аналізу. Застосування таких методів є критично важливим, оскільки мережеві потоки можуть містити сотні параметрів, що ускладнює роботу моделей без попереднього препроцесингу [12].

Машинне навчання у контексті мережевої безпеки базується також на правильному формуванні ознак для моделі. Серед найпоширеніших ознак, які використовуються у IDS, можна назвати кількість пакетів за одиницю часу, розмір пакетів, тривалість з’єднання, кількість встановлених сесій для певного порту, напрямок трафіку, статистичні характеристики інтервалів між пакетами та типи протоколів. Чим більш інформативно сформована матриця ознак, тим ефективніше працює модель [11].

Таким чином, сучасні алгоритми машинного навчання надають широкий інструментарій для побудови ефективних систем виявлення вторгнень. Їх здатність адаптуватися до нових патернів трафіку, обробляти великі обсяги даних та забезпечувати високоточні результати робить машинне навчання незамінним елементом сучасних IDS [13].

2.2 Застосування машинного навчання у сфері кібербезпеки

Сфера кібербезпеки сьогодні характеризується динамічним розвитком, збільшенням кількості атак та їх складністю. Злочинці активно використовують автоматизацію, інтелектуальні скрипти та обхідні техніки для порушення безпеки інформаційних систем. У такому середовищі традиційні методи виявлення загроз стають недостатніми, адже базуються на заздалегідь визначених правилах та сигнатурах, які швидко застарівають. Машинне навчання стало основою нового покоління інструментів безпеки, здатних адаптуватися до змін у поведінці користувачів, обладнання та атакувальників.

Одним із ключових застосувань машинного навчання в кібербезпеці є виявлення аномалій [14]. На відміну від сигнатурних методів, аномалійні алгоритми аналізують характеристики нормального трафіку й виявляють будь-які відхилення, що можуть сигналізувати про зловмисну активність. Це робить їх особливо ефективними для виявлення нових атак нульового дня, шкідливих дій інсайдерів, а також складних багатостадійних вторгнень, які традиційні системи не можуть розпізнати. Моделі машинного навчання забезпечують можливість створення профілів поведінки користувачів та пристроїв, що дозволяє виявляти відхилення у звичному функціонуванні системи.

Застосування машинного навчання також поширюється на аналіз логів, потокових даних, NetFlow-записів і системних подій. Використання глибинного аналізу дозволяє автоматично класифікувати події за рівнем ризику, визначати взаємозв’язки між різними інцидентами та формувати

комплексну картину стану безпеки мережі [14]. Це важливо для сучасних SOC, де обсяг інцидентів може перевищувати можливості ручної обробки.

Крім того, машинне навчання активно застосовується у виявленні шкідливих програм. Моделі, натреновані на великій кількості прикладів файлів, можуть розпізнавати шкідливі виконавчі файли навіть тоді, коли вони модифіковані або обфусковані. Глибокі нейронні мережі, здатні аналізувати внутрішню структуру виконуваних файлів, працюють значно ефективніше, ніж класичні антивірусні механізми, що базуються на сигнатурах. Це дозволяє значно підвищити швидкість реагування на шкідливі кампанії та мінімізувати ризики зараження.

У сфері аналізу мережевого трафіку машинне навчання використовується для класифікації типів трафіку, визначення підозрілих шаблонів комунікацій, класифікації та пріоритизації інцидентів, а також аналізу зашифрованого трафіку. Зі збільшенням поширення протоколів шифрування, таких як TLS 1.3, можливості традиційних IDS значно зменшуються, адже контент пакетів стає недоступним. У таких ситуаціях машинне навчання дозволяє аналізувати метадані трафіку, включаючи його часові характеристики, розмір пакетів та інші непрямі ознаки, що дає змогу визначати, чи є сесія потенційно шкідливою [15].

Ще одним напрямом є використання ML у прогнозуванні кіберзагроз. Статистичні моделі та рекурентні нейронні мережі здатні прогнозувати майбутню активність атак на основі історичних даних. Це забезпечує можливість проактивного захисту, коли система може запобігати атакам ще до того, як вони будуть розгорнуті зловмисниками. Такий підхід є одним із ключових компонентів концепції кібербезпеки нового покоління [16].

Отже, машинне навчання стало невід’ємною частиною сучасних рішень у сфері кіберзахисту. Його застосування дозволяє автоматизувати складні процеси, підвищувати точність виявлення загроз, знижувати кількість хибнопозитивних спрацювань і суттєво покращувати загальну ефективність захисних систем.

2.3 Методи використання нейронних мереж для аналізу атак

Нейронні мережі стали одним із найбільш потужних і гнучких інструментів для побудови систем виявлення вторгнень, здатних працювати з великими обсягами даних і складними нелінійними залежностями. На відміну від класичних моделей машинного навчання, нейронні мережі можуть моделювати складні взаємозв'язки між ознаками, що є особливо важливим для мережевого трафіку, який має варіативну структуру, різноманітні часові закономірності та велику кількість потенційних шаблонів поведінки [16].

Одним із найбільш ефективних методів використання нейронних мереж у виявленні вторгнень є застосування багатошарових перцептронів. Ці мережі здатні виконувати багатокласову класифікацію, розрізняючи різні типи атак на основі великої кількості ознак. Використання перцептронів є простішим, ніж робота з більш складними архітектурами, але водночас вони забезпечують високу точність за умови правильного налаштування параметрів. Недоліком таких мереж є те, що вони не враховують часовий контекст, що є важливим для мережевого аналізу[13].

Для роботи з часовими рядами трафіку особливо корисними є рекурентні нейронні мережі, зокрема LSTM та GRU. Вони дозволяють аналізувати залежності між подіями у часі та визначати аномалії, які розвиваються поступово. Наприклад, повільні DDoS-атаки або атаки типу port scanning мають характерні часові патерни, які можуть бути невидимими для класичних моделей. Рекурентні мережі здатні моделювати довготривалі залежності та виявляти підозрілі серії подій, що робить їх незамінними для потокового аналізу мережі [17].

Ще одним важливим напрямом є використання згорткових нейронних мереж для аналізу зашифрованого та незашифрованого трафіку. Хоча згорткові мережі традиційно застосовувалися у сфері комп'ютерного зору, їх можна ефективно використовувати для обробки структурованих даних, включаючи послідовності мережевих пакетів. Вони можуть виділяти локальні патерни у даних, які відповідають характерним ознакам атак, таким як

аномальні розподіли розмірів пакетів, частота передачі або специфічна послідовність TCP-прапорців. Дослідження показали, що CNN можуть досягати високої точності у виявленні складних атак, особливо при використанні гібридних моделей [16].

Глибокі автоенкодери є ще одним інструментом, який широко застосовується у сфері IDS. Автоенкодери дозволяють виявляти аномалії шляхом навчання компактного представлення нормального трафіку. Після тренування на легітимних даних автоенкодер намагається реконструювати вхідні дані [15]. Якщо подані дані суттєво відрізняються від нормальних, помилка реконструкції зростає, що дає змогу використовувати автоенкодер як детектор аномалій. Це особливо корисно для виявлення нових атак, для яких немає заздалегідь підготовлених сигнатур чи навчальних вибірок.

Також важливим напрямом є використання генеративно-змагальних мереж (GAN). Вони можуть застосовуватися для формування синтетичних даних, що допомагає вирішувати проблему дисбалансу класів у навчальних вибірках. Оскільки атаки зустрічаються значно рідше, ніж легітимний трафік, моделі часто схильні «ігнорувати» рідкісні події. GAN дозволяють створювати реалістичні приклади атакувального трафіку, полегшуючи навчання моделей і збільшуючи загальну точність виявлення [17].

Крім того, нейронні мережі активно застосовуються у задачах кореляції інцидентів та побудови моделей поведінки. Використання глибинних підходів дозволяє поєднувати дані з кількох джерел, включаючи логи IDS, системні події, інформацію про робочі станції, мережеві потоки та інші елементи інфраструктури [13]. Це дозволяє створити комплексну картину атаки й автоматизувати процес ухвалення рішень щодо реагування.

Таким чином, нейронні мережі забезпечують потужний інструментарій для побудови інтелектуальних систем виявлення вторгнень. Їх здатність працювати з великими обсягами складноструктурованих даних, аналізувати часові закономірності та знаходити приховані залежності робить їх ключовим компонентом IDS нового покоління.

2.4 Використання методу Random Forest для аналізу логів

Метод Random Forest є одним із найбільш практичних і широко застосовуваних алгоритмів машинного навчання для задач класифікації та регресії, і його властивості роблять його особливо придатним для аналізу мережевих логів. У найзагальнішому вигляді Random Forest — це ансамбль незалежних дерев рішень, об’єднаних у сумарну модель шляхом голосування (для задач класифікації) або середнього значення (для регресії) [9]. Ідея ансамблю полягає в тому, що хоча окреме дерево може бути схильним до перенавчання та дуже чутливим до шуму у даних, незалежне тренування великої кількості дерев на різних підвибірках даних і підмножинах ознак дозволяє зменшити дисперсію моделі та підвищити стабільність і узагальнювальні властивості результату. У контексті аналізу логів це означає, що Random Forest здатний ефективно працювати з великою кількістю різнорідних ознак, нечітких та частково корельованих характеристик, витримувати присутність відсутніх значень або шуму, а також давати інтерпретовані індикації про важливість ознак, що робить його корисним інструментом у проєктах інформаційної безпеки [9].

На рисунку 2.1 показано, як працює алгоритм Random Forest у спрощеному вигляді [9]. Один і той самий вхідний об’єкт (instance) подається на кілька різних дерев рішень. Кожне дерево, навчене на своїй випадковій підвибірці даних та ознак, приймає власне рішення — наприклад, Tree-1 дає клас A, Tree-2 та Tree-n дають клас B. Після цього всі отримані результати об’єднуються за принципом більшості голосів (majority voting). Клас, який отримав найбільшу кількість «голосів» від дерев, і стає фінальним прогнозом моделі.

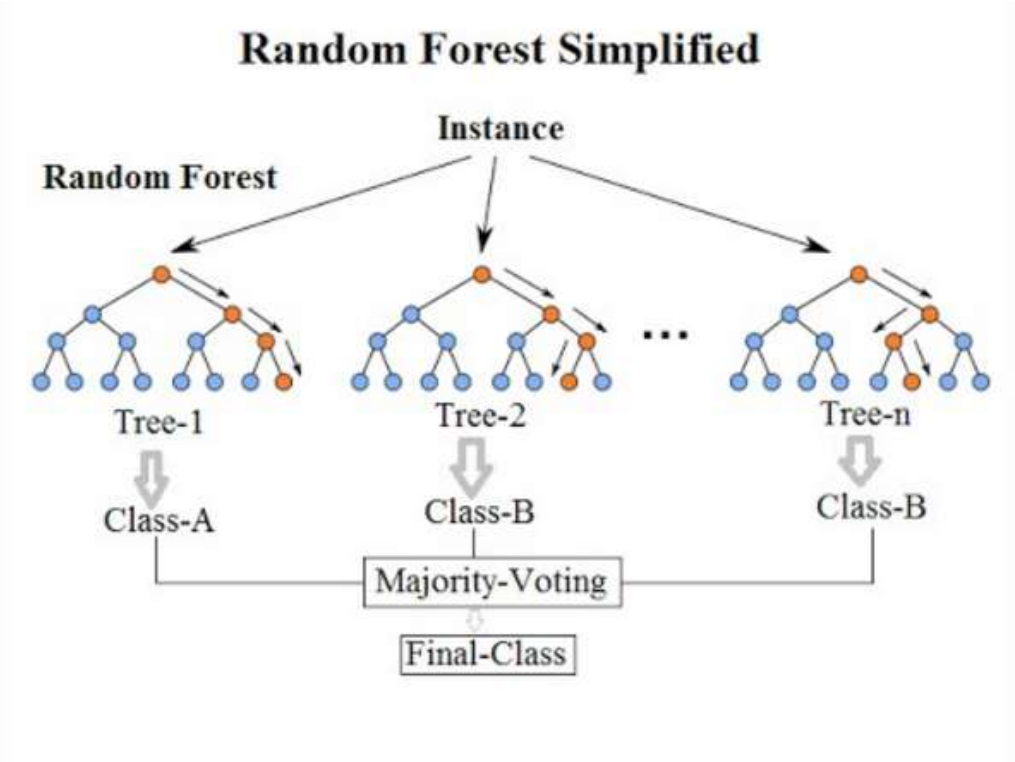


Рисунок 2.1 – Схема побудови Random Forest

Технічно кожне окреме дерево в лісі будується за алгоритмом побудови дерева рішень: на кожному вузлі обирається найкраще порогове розбиття по одній з доступних ознак за критерієм чистоти (часто використовують критерії ентропії або Gini). Random Forest додає до цього стандартного підходу два джерела стохастичності [12]. Перше — це бэггінг (bootstrap aggregation): для кожного дерева формується своя навчальна вибірка шляхом випадкової вибірки з поверненням з початкового датасету, тобто кожне дерево бачить приблизно 63.2% унікальних прикладів оригінального набору і певну кількість повторів. Друге джерело стохастичності полягає у випадковому відборі підмножини ознак при кожному розбитті вузла: замість розгляду всього набору ознак для пошуку найкращого розбиття дерево обирає k випадкових ознак і знаходить серед них оптимальний поріг. Ця випадкова підстановка ознак забезпечує додаткову декореляцію між деревами, що ще більше зменшує дисперсію ансамблю. Сумарний прогноз отримується через агрегування індивідуальних прогнозів: у задачі класифікації шляхом більшості голосів, в задачі регресії — середнім значенням прогнозів усіх дерев [9].

Для застосування Random Forest до аналізу логів важливо розуміти характер вхідних даних і етапи підготовки, без яких навіть потужний ансамбль виявиться малоефективним. Мережеві логи, включно з форматом Suricata eve.json, Zeek-логами або NetFlow-експортом, мають ряд особливостей: велика кількість рядків (потоки подій), різноманітні типи полів (категоріальні, числові, часові, текстові), нерівномірний розподіл класів (наприклад, дуже мало атак порівняно з нормальним трафіком), пропуски у значеннях та наявність дублювань [12]. Перший крок у роботі — побудова конвеєра передобробки даних. Він включає нормалізацію часових міток, агрегацію пакетів у сесії або потоки, виведення часових та статистичних ознак, перетворення текстових полів у числові ознаки, обробку пропусків і кодування категоріальних змінних. Конкретний набір ознак визначається доступними логами та цілями аналізу: для виявлення DDoS-атак корисними будуть ознаки, що описують швидкість потоку, кількість унікальних IP-адрес за короткий інтервал, співвідношення вхідних та вихідних пакетів; для виявлення експлойтів важливими стануть індикатори аномалій у сигнатурах протоколів, невідповідність полів заголовків або раптові зміни розмірів пакетів. Критично важливо звести до єдиного формату часові мітки та часові інтервали, оскільки багато алгоритмів базуються на агрегації подій у часових вікнах. Аггрегація може бути скользячою або фіксованою по інтервалах; вибір залежить від характеру атак, які прагнуть виявити.

Підготовка ознак починається зі структуризації сирих логів. Наприклад, із Suricata eve.json для кожного запису можна вичленувати поля типу timestamp, src_ip, dest_ip, src_port, dest_port, proto, http.request.uri, http.user_agent, flow.duration, flow.bytes_toserver, flow.bytes_toclient, alert.signature, alert.category та інші [18]. Після цього логічно агрегувати потоки по 5-секундних або 1-хвилинних інтервалах для отримання характеристик потоку: сумарна кількість пакетів, середній розмір пакету, кількість різних портів, число унікальних джерел, частота підключень до того самого сервісу, частота невдалих спроб аутентифікації тощо. Часові ознаки, такі як час доби

або день тижня, можуть додавати значущу інформацію, оскільки деякі аномалії частіше трапляються у неробочий час. Для категоріальних полів застосовується кодування: просте one-hot кодування підходить для невеликої кількості унікальних значень, тоді як для великих простірів значень (наприклад, user-agent) застосовують техніки частотного кодування або інкорпорацію embeddings [9]. Текстові поля, наприклад URL або рядки підписів, можна перетворювати у векторні представлення за допомогою TF-IDF або більш досконалих методів ембеддінгів, однак слід пам’ятати, що Random Forest найкраще працює з табличними числовими ознаками, тому після перетворення тексту бажано скоротити вимірність векторів до керованого розміру через методи відбору ознак або зменшення розмірності.

Проблема незбалансованості класів у логах є однією з ключових практичних складностей. У типовому наборі даних частка реальних атак може складати менше одного відсотка. Random Forest має природну стійкість до деякого дисбалансу, але ефективність детектора підвищується при застосуванні додаткових прийомів: повторна вибірка меншого класу (oversampling), відсікання частини більшого класу (undersampling), або використання алгоритмів генерації синтетичних прикладів, наприклад SMOTE [12]. Кожен із цих підходів має свої обмеження: oversampling підвищує ризик перенавчання, undersampling може втратити важливу інформацію про нормальний трафік, а SMOTE може створювати неприродні приклади у випадках сильно нерегулярного розподілу ознак. Практична стратегія часто є комбінованою: попередня селекція ознак, застосування SMOTE лише до підмножини ознак, контроль за узгодженістю генерації синтетичних прикладів, а також валідація на окремому наборі реальних даних [9].

Навчання Random Forest супроводжується вибором низки гіперпараметрів, які суттєво впливають на продуктивність моделі. Серед найважливіших — кількість дерев у лісі (n_estimators), максимальна глибина окремого дерева (max_depth), мінімальна кількість зразків у вузлі для поділу (min_samples_split), мінімальна кількість зразків у листі (min_samples_leaf),

число ознак, розглянутих при кожному розбитті (max_features), критерій якості поділу (наприклад, Gini або ентропія) та параметри для обмеження розростання дерева [9]. Збільшення кількості дерев зазвичай покращує узагальнення за ціною збільшення часу тренування та вимог до пам’яті; після певного значення прибуток від додаткових дерев стає незначним. Обмеження глибини дерева і мінімальної кількості зразків в листі контролює схильність до перенавчання: менші дерева більш узагальнювальні, проте можуть не вловити складні залежності. Вибір оптимального набору гіперпараметрів зазвичай здійснюється методами крос-валідації: можна застосовувати сітковий пошук (grid search), випадковий пошук (random search) або більш просунуті байєсові підходи оптимізації, які дозволяють знайти збалансоване поєднання точності і обчислювальної ефективності. Для великих наборів даних практичним є початкове обмеження розміру пошукового простору та поступове тонке налаштування на менших підвибірках [12].

Оцінка якості моделі в задачах виявлення вторгнень повинна враховувати не лише загальну точність, яка у випадку незбалансованих наборів може бути оманливою, а й метрики, орієнтовані на класи: precision, recall, F1-score, area under ROC curve (AUC-ROC), area under Precision-Recall curve (AUC-PR). Для безпеки особливо важливий recall (чутливість), оскільки пропуск реальної атаки часто має серйозні наслідки. Водночас низький precision призводить до великої кількості хибних спрацьовувань, що перевантажує аналітичні підрозділи. Тому оптимальним є баланс між цими метриками, який залежить від організаційної стратегії реагування на інциденти [19]. У практичних впровадженнях застосовують політику багаторівневої реакції: модель із високим recall і середнім precision може використовуватися для попереднього скринінгу, а подальша корекція і фільтрація інцидентів здійснюється іншими механізмами або ручною перевіркою.

Для інтерпретації моделі Random Forest має цілий набір корисних інструментів. Стандартним вихідним результатом є оцінки важливості ознак,

які можна отримати шляхом вимірювання зменшення критерію чистоти при поділах, до яких привела конкретна ознака [9]. Такі глобальні показники важливості дають уявлення про те, які ознаки найчастіше використовуються деревами при побудові рішень. Однак ця метрика може бути зміщеною у бік ознак з більшою кількістю можливих розбиттів або тих, що мають сильний корельований зв'язок з іншими. Для отримання більш надійних інсайтів застосовують методи на кшталт permutation importance, які вимірюють падіння якості моделі при випадковому перемішуванні значень конкретної ознаки [18]. Сучасні практики інтерпретації включають використання локальних методів, таких як LIME або SHAP, які дозволяють оцінити внесок кожної ознаки до конкретного прогнозу. Для SOC це корисно, оскільки інтерпретовані причини спрацювання допомагають аналітикам швидше розуміти природу інциденту та приймати рішення про подальші дії.

Практичне розгортання Random Forest в продуктивному середовищі має свої особливості. По-перше, модель має бути інтегрована в pipeline, що обробляє реальний потік логів у близькому до реального часу. Це означає, що етапи попередньої обробки, які застосовувались при навчанні (нормалізація, кодування, агрегація), повинні бути відтворені в продуктивному коді у тій же послідовності [13]. Часто для цього створюють набір серіалізованих трансформерів або використовується формат збереження моделі разом із препроцесором (наприклад, через joblib, pickle або власні сервіси міксинів для трансформацій). Друге — швидкість прогнозування. Random Forest зазвичай забезпечує високу швидкість інференсу порівняно з важкими нейронними мережами, але при великій кількості дерев і високій розмірності ознак час прогнозування може бути немалий. У таких випадках корисні оптимізації: зменшення кількості дерев, обмеження глибини, використання компактних представлень ознак, або застосування спеціалізованих реалізацій (наприклад, реалізації на C/C++ або бібліотеки, що використовують паралельну обробку). Третє — моніторинг деградації моделі [9]. Мережеве середовище змінюється: патерни трафіку еволюціонують, нові сервіси з'являються, а старі змінюють

поведінку. Тому необхідно регулярно відстежувати показники якості моделі на поточних даних і періодично проводити її перевчання або донавчання на нових анотаціях. Автоматизація циклу MLOps, що включає збір нових підписаних інцидентів, оновлення датасету, перекваліфікацію моделі та її деплой у production, суттєво підвищує стійкість системи [17].

Особливу увагу слід приділити роботі з ознаками, що можуть містити витік інформації або не бути доступними у продуктивному середовищі. Наприклад, під час побудови моделі для експериментального набору даних могли використовуватися ознаки, які залежать від майбутніх подій або з’являються лише після часткової обробки (target leakage). Такі ознаки треба виявляти і виключати, інакше модель покаже завищену продуктивність у тестах, але провалиться у реальній експлуатації. Для цього корисні процедурні контролі на етапі інженерії ознак: чітке відділення часових вікон для тренування та тестування, використання only-past features для прогнозування майбутніх подій і ревізія логіки агрегації, щоб не випадково включити інформацію з майбутнього [15].

Реальний приклад практичного workflow із застосування Random Forest для аналізу Suricata eve.json виглядає як послідовність етапів, які плавно перетікають один в інший. Спершу визначаються сценарії атак і формуються цільові мітки (labels), що може вимагати ручної розмітки або поєднання правил кореляції з експертними знаннями. Далі виконується екстракція ознак: для кожного мережевого потоку або агрегаційної одиниці обчислюються статистики, часові індикатори, частотні та категоріальні індикатори. Після цього дані очищаються від пропущених значень, категоріальні поля кодуються, числові нормалізуються, і проводиться селекція ознак з використанням кореляційного аналізу, тестів важливості або методів відбору ознак, таких як Recursive Feature Elimination [9]. Потім формується навчальна та тестова вибірки за принципом часової розбивки, аби уникнути проникнення майбутньої інформації у навчання. Модель Random Forest тренується, відбувається налаштування гіперпараметрів через крос-валідацію, після чого

її якість оцінюється на відкладеному ще не баченому наборі. Результати оцінки демонструються у вигляді матриць плутанини, ROC- та PR-кривих, а також метрик за класами [13]. На етапі інтерпретації підраховується важливість ознак і аналізуються топознаки, які приводять до спрацьовувань. При задовільних показниках формується pipeline для продуктивного деплою з серіалізацією моделі та трансформерів, а також налаштовується моніторинг якості й логування прогнозів і причин спрацьовування для подальшого аналізу та донавчання.

Важливо також говорити про обмеження Random Forest і випадки, коли цей метод може бути менш ефективним. Random Forest не забезпечує найкращої продуктивності при екстремально розріджених та надвисокорозмірних текстових вхідних просторах без попереднього зменшення розмірності [10]. Для задач типу аналізу raw payload або складної обробки послідовностей пакетів іноді краще підходять спеціалізовані архітектури на основі нейронних мереж (LSTM, CNN) або трансформери. Random Forest також важче використовувати для задач, що вимагають прогнозування у реальному часі із затримками у межах мілісекунд у масштабах великих мереж, якщо модель має сотні дерев і велике число ознак; у таких випадках потрібно або оптимізувати модель, або розглядати розподілені обчислення. Проте переваги в інтерпретованості, простоті впровадження та стійкості до різних типів шуму роблять Random Forest відмінним вибором як базовий або другий рівень аналізу у багаторівневих системах виявлення вторгнень [9].

Нарешті, аспект безпеки та етичні питання також важливі у впровадженні. При побудові моделі слід гарантувати, що використання приватних або персональних даних у логах відповідає політикам конфіденційності та законодавству. Для цього застосовуються методи анонімізації IP-адрес, видалення персональних ідентифікаційних полів або трансформації їх у агреговані статистики. Паралельно варто впроваджувати контроль доступу до моделі та журналів прогнозів, забезпечувати аудит дій і

можливість пояснення спрацювань для внутрішнього контролю або судового розгляду [19].

У підсумку, Random Forest є потужним, гнучким і практично орієнтованим методом для аналізу мережевих логів. Його ключові переваги полягають у стійкості до шуму, здатності працювати з різними типами ознак, інтерпретованості та порівняно невеликій складності розгортання. Для досягнення найкращих результатів у задачах IDS важливо створити якісний pipeline передобробки даних, правильно інженерити ознаки, коригувати дисбаланс класів, ретельно підбирати гіперпараметри, проводити валідацію за часовим принципом, інтерпретувати результати через сучасні методи пояснення і впроваджувати модель у продуктивне середовище з моніторингом деградації й механізмами донавчання. Тільки такий комплексний підхід дозволить ефективно застосувати Random Forest для виявлення вторгнень у мережі та підвищити загальний рівень кібербезпеки організації.

3 МОДЕЛЮВАННЯ ТА АНАЛІЗ СИСТЕМИ ВИЯВЛЕННЯ ВТОРГНЕНЬ

3.1 Побудова та налаштування мережі для тестування моделі

Для проведення тестування було використано програму Hyper-V Manager, для побудови мережі, що включає в себе такі віртуальні машини:

- Атакуючої Kali Linux машини за IP адресою: 172.31.136.82;
- IDS системою Linux на основі Suricata з IP адресою: 172.31.133.207;
- Та цілі Windows 10, на яку будуть проводитися атаки, з IP адресою: 172.31.137.98;

Загальна структура топології мережі зображена на рисунку 3.1 Ключовим елементом топології був віртуальний комутатор, налаштований на дзеркалювання трафіку (SPAN/Mirror port). Це забезпечувало IDS Suricata копією всього трафіку, що проходив між атакуючою машиною і ціллю. Така конфігурація дозволила моніторити трафік без внесення затримок у його передачу та коректно збирати дані для подальшого аналізу.

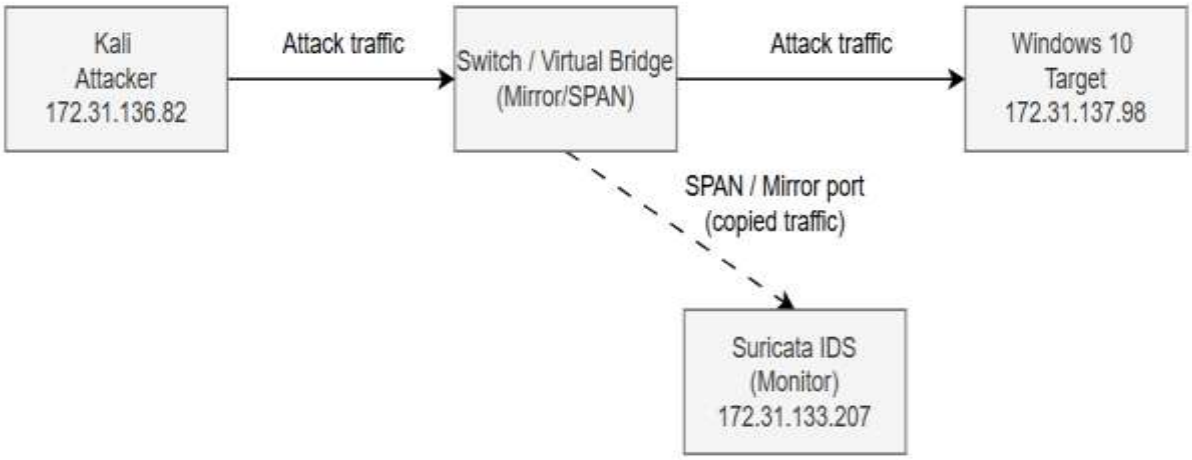


Рисунок 3.1 – Топологія мережі

На рисунках 3.2 – 3.4 наведено мережеву конфігурацію створених віртуальних машин.

```
C:\Windows\system32>ipconfig

Настройка протокола IP для Windows


Адаптер Ethernet Ethernet:

   Состояние среды. . . . . . . . : Среда передачи недоступна.
   DNS-суффикс подключения . . . . . : local

Адаптер Ethernet Ethernet 2:

   DNS-суффикс подключения . . . . . : mshome.net
   Локальный IPv6-адрес канала . . . : fe80::4e66:bb30:6b22:2024%6
   IPv4-адрес. . . . . . . . . . . . : 172.31.137.98
   Маска подсети . . . . . . . . . . : 255.255.240.0
   Основной шлюз. . . . . . . . . : 172.31.128.1
```

Рисунок 3.2 – IP адреса цільової віртуальної машини Windows

```
maksym@maksym-Virtual-Machine: $ ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 172.31.133.207  netmask 255.255.240.0  broadcast 172.31.143.255
        inet6 fe80::89f1:352c:f593:e4d  prefixlen 64  scopeid 0x20<link>
        ether 00:15:5d:01:11:0a  txqueuelen 1000  (Ethernet)
        RX packets 109341  bytes 142783835 (142.7 MB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 8247  bytes 605248 (605.2 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

eth1: flags=4099<UP,BROADCAST,MULTICAST>  mtu 1500
        ether 00:15:5d:01:11:0b  txqueuelen 1000  (Ethernet)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

Рисук 3.3 – IP адреса віртуальної машини з IDS

```
┌──(kali㉿kali)-[~]
└─$ ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 172.31.136.82  netmask 255.255.240.0  broadcast 172.31.143.255
        inet6 fe80::c601:4b5c:2747:9d28  prefixlen 64  scopeid 0x20<link>
        ether 00:15:5d:01:11:02  txqueuelen 1000  (Ethernet)
        RX packets 194  bytes 42192 (41.2 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 55  bytes 12470 (12.1 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 8  bytes 480 (480.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 8  bytes 480 (480.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

Рисунок 3.4 – IP адреса атакуючої машини Kali Linux

Створені віртуальні машини було підключено до спільної мережі та перевірено ICMP з'єднання між ними за допомогою утиліти ping, як вказно на рисунках 3.5-3.7. Наступним кроком була синхронізація часу на усіх машинах для точного визначення часових рамок атак та сповіщень безпеки.

```
C:\Windows\system32>ping 172.31.136.82

Обмен пакетами с 172.31.136.82 по с 32 байтами данных:
Ответ от 172.31.136.82: число байт=32 время<1мс TTL=64
Ответ от 172.31.136.82: число байт=32 время<1мс TTL=64
Ответ от 172.31.136.82: число байт=32 время<1мс TTL=64
Ответ от 172.31.136.82: число байт=32 время<1мс TTL=64

Статистика Ping для 172.31.136.82:
    Пакетов: отправлено = 4, получено = 4, потеряно = 0
    (0% потерь)
Приблизительное время приема-передачи в мс:
    Минимальное = 0мсек, Максимальное = 0 мсек, Среднее = 0 мсек

C:\Windows\system32>ping 172.31.133.207

Обмен пакетами с 172.31.133.207 по с 32 байтами данных:
Ответ от 172.31.133.207: число байт=32 время<1мс TTL=64
Ответ от 172.31.133.207: число байт=32 время<1мс TTL=64
Ответ от 172.31.133.207: число байт=32 время<1мс TTL=64
Ответ от 172.31.133.207: число байт=32 время<1мс TTL=64

Статистика Ping для 172.31.133.207:
    Пакетов: отправлено = 4, получено = 4, потеряно = 0
    (0% потерь)
Приблизительное время приема-передачи в мс:
    Минимальное = 0мсек, Максимальное = 0 мсек, Среднее = 0 мсек
```

Рисунок 3.5 – Перевірка з'єднання цільової віртуальної машини Windows

```
maksym@maksym-Virtual-Machine:~$ ping 172.31.136.82
PING 172.31.136.82 (172.31.136.82) 56(84) bytes of data.
64 bytes from 172.31.136.82: icmp_seq=1 ttl=64 time=0.255 ms
64 bytes from 172.31.136.82: icmp_seq=2 ttl=64 time=0.419 ms
64 bytes from 172.31.136.82: icmp_seq=3 ttl=64 time=0.412 ms
64 bytes from 172.31.136.82: icmp_seq=4 ttl=64 time=0.445 ms

^C--- 172.31.136.82 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3096ms
rtt min/avg/max/mdev = 0.255/0.382/0.445/0.074 ms
maksym@maksym-Virtual-Machine:~$ ping 172.31.137.98
PING 172.31.137.98 (172.31.137.98) 56(84) bytes of data.
64 bytes from 172.31.137.98: icmp_seq=1 ttl=128 time=0.529 ms
64 bytes from 172.31.137.98: icmp_seq=2 ttl=128 time=0.457 ms
64 bytes from 172.31.137.98: icmp_seq=3 ttl=128 time=0.412 ms
64 bytes from 172.31.137.98: icmp_seq=4 ttl=128 time=0.588 ms
^C
--- 172.31.137.98 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3077ms
rtt min/avg/max/mdev = 0.412/0.496/0.588/0.067 ms
```

Рисунок 3.6 – Перевірка з'єднання віртуальної машини з IDS

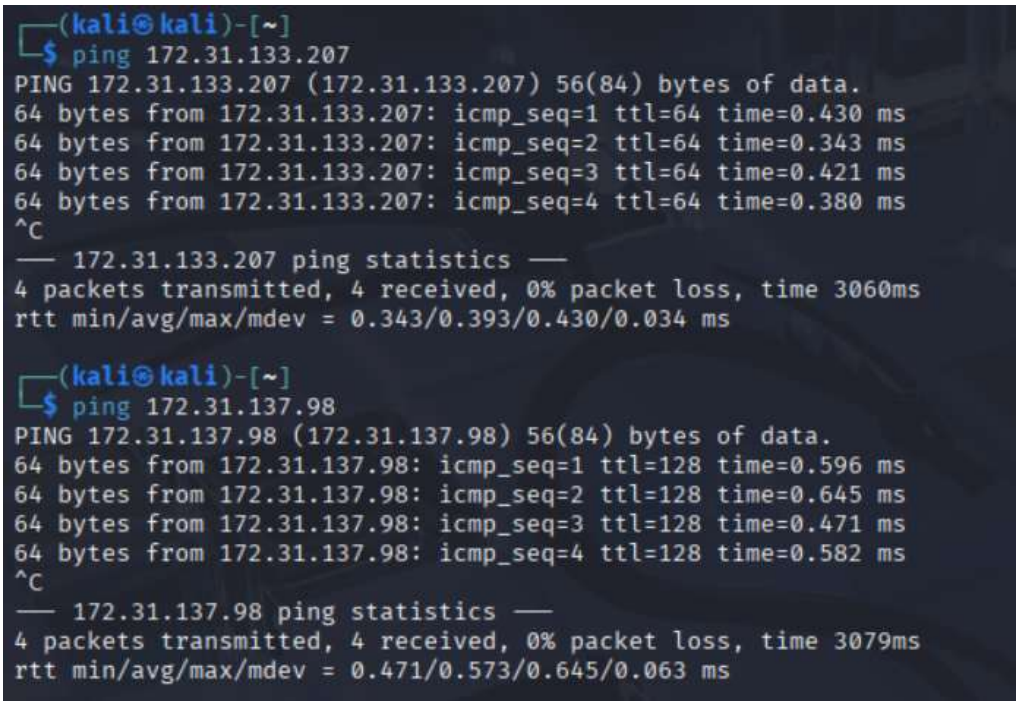


Рисунок 3.7 – Перевірка з'єднання атакуючої машини Kali Linux

Після налаштування мережі переходимо до встановлення системи вивлення вторгнення IDS Suricata. Для цього було завантажено останню версію з GitHub та перевірено чи вдало встановлено командою перевірки статусу, як вказано на малюнку 3.8.

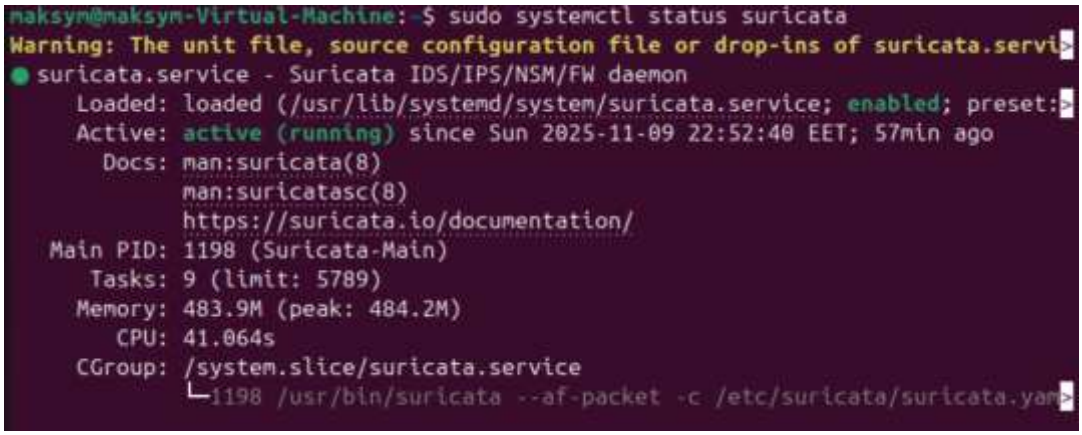


Рисунок 3.8 – Вдале встановлення IDS Suricata

Також треба переконатися, що IDS працює на правильному інтерфейсі та для виводу обрано саме потрібний нам файл, за замовчуванням усі логи записуються у eve.json файл. На рисунку 3.9 продемонстровано налаштування інтерфейсу, а на рисунку 3.10 вказано де будуть зберігатися сповіщення та які типи буде реагувати IDS.

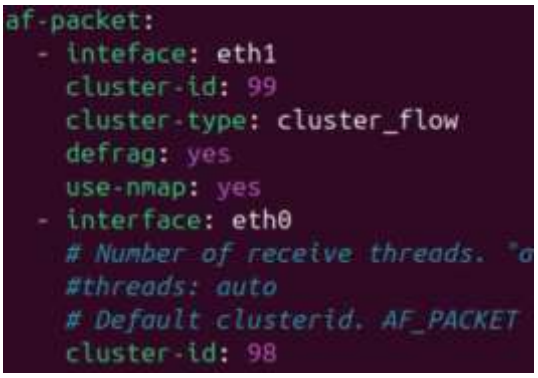


Рисунок 3.9 – Налаштування інтерфейсів

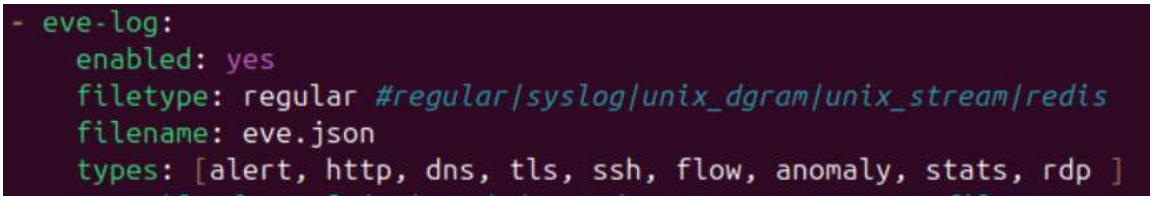


Рисунок 3.10 – Налаштування виводу

Налаштування інтерфейсу та файлу виводу встановлені, тепер переходимо до створення локальних правил, щоб було наглядніше видно роботу IDS під час атак. Загальні правила для сповіщень безпеки зберігаються у файлі /var/lib/suricata/rules/suricata.rules. Щоб додати локальні правила до IDS Suricata будо створено у директорії, де знаходсяться основні правила, тобто у /var/lib/suricata/rules файл local.rules. Сам файл було заповнено правилами, що дають змогу виявити незначні атаки на мережу. Зміст файлу вказано у Додатку А.1.

Марежа налаштована, та готова до використання.

3.2 Вибір та підготовка даних для навчання моделей

У процесі розроблення системи виявлення вторгнень на основі поєднання класичних сигнатурних методів та алгоритмів машинного навчання особливе значення має коректний вибір і підготовка даних, на яких буде здійснюватися навчання моделі. Саме якість вихідних наборів даних, їх різноманітність, репрезентативність для реального середовища та рівень попереднього опрацювання визначають, наскільки точною, стабільною та ефективною буде модель у подальших експериментах [20]. У рамках цього дослідження для формування навчального набору використовувалися як

трафік, згенерований у власній лабораторній мережі, так і відкриті набори даних, отримані з репозиторіїв GitHub, які містять як приклади нормального трафіку, так і різноманітні типи атак, що охоплюють широке коло протоколів, методик компрометації та патернів шкідливої поведінки [21].

Початковим етапом є формування експериментальної мережі, яка складається з атакуючої машини Kali Linux, цільової Windows 10 та системи IDS/IPS Suricata, що працює під управлінням Linux. Усі вузли під'єднані через віртуальний мережевий комутатор, що дозволяє створити середовище, максимально наближене до реального мережевого сегменту, але водночас контрольоване й безпечне. Трафік між Kali Linux та Windows 10 дублюється на інтерфейс Suricata за допомогою функції SPAN/port mirroring, що забезпечує можливість захоплення всіх пакетів і подальшого формування журналів подій у форматі eve.json. Саме ці журнали й виступають вихідним матеріалом для подальшого аналізу, попередньої обробки та перетворення у формат, придатний для навчання моделі.

Після налаштування мережі і запуску Suricata виконується серія тестових атак, які включають мережеве сканування, ICMP-флуди, DoS-атаки, спроби брутфорс-автентифікації, небезпечні HTTP-запити, експлуатацію вразливостей, спроби встановлення SSH-з'єднань, а також інші сценарії, характерні для типових загроз. Мета цього етапу полягає у створенні максимально різноманітного набору прикладів, які демонструють аномальну поведінку в мережі й забезпечують моделі можливість навчитися відрізняти нормальний трафік від шкідливого [22]. Усі події, згенеровані Suricata під час атак, автоматично записуються у файл eve.json у структурованому форматі JSON. Це дає змогу легко аналізувати події, витягувати необхідні поля та поєднувати їх з інформацією про час, IP-адреси, типи протоколів, сигнатури, які вони спровокували, та інші атрибути.

Нижче подано опис атак які було використано для генерації шкідливого трафіку.

1) Nmap SYN Scan (швидке сканування портів)

nmap -sS -p1-2000 -T4 172.22.242.178

Цей тест генерує великий обсяг напіввідкритих TCP SYN-з'єднань (так званий "SYN stealth scan"). Атакуючий хост відправляє SYN-пакети на діапазон портів 1–2000, але не завершує встановлення TCP-сесії. Такий метод широко використовується під час первинної розвідки для визначення відкритих портів і працюючих служб. Для IDS він виглядає як аномальна кількість SYN-пакетів без належного завершення handshake, тому має бути класифікований як порт-сканування [10].

2) Брутфорс логінів через користувацький скрипт

./scripts/attack_bruteforce.sh 172.22.242.178 ~/lists/test_us.txt ~/lists/test_ps.txt

Скрипт автоматично перебирає можливі пари логін/пароль зі списків і намагається встановити сеанс автентифікації з ціллю. Атака імітує спроби підбору облікових даних на одному з сервісів, доступних на сервері. Такий трафік характеризується великою кількістю коротких повторюваних спроб підключення, що часто генерує серію схожих IDS-алертів [6].

3) Hydra brute-force на RDP

hydra -t 4 -f -V -l maxbl -P ~/lists/test_ps.txt rdp://172.22.242.178

Hydra виконує швидку багатопоточну атаку перебору пароля до протоколу RDP. Інструмент посилає численні послідовні запити автентифікації, змінюючи пароль при кожній спробі. Для IDS така поведінка є типовим патерном вертикального brute-force: один логін, багато паролів, висока інтенсивність сесій, повторювані спроби з того самого джерела [2].

4) SQL-ін'єкція (SQLi-like HTTP-атака)

curl -v "http://172.22.242.178:8080/search?q=%27%20OR%201=1%20--" for i in {1..20}; do curl -s" http://172.22.242.178:8080/search?q=%27%20OR%201=1%20--" >/dev/null; done

Це один із базових варіантів SQL-ін'єкції, у якому параметр запиту модифікується шляхом вставки виразу ' OR 1=1 --. Такий запит намагається

змусити сервер повернути всі записи або виконати некоректну SQL-команду. Повторення запиту 20 разів створює характерний HTTP-трафік, який легко ідентифікується IDS як спроба SQLi чи шкідлива активність на веб-додатку [8].

5) DoS SYN-flood через hping3

sudo hping3 --flood -S -p 80 172.22.242.178

Атака SYN-flood генерує масовий потік SYN-пакетів на порт 80 без завершення TCP-з'єднання. Це створює навантаження на стек TCP-цілі і може призвести до відмови в обслуговуванні. З точки зору IDS, така активність виглядає як високочастотний однотипний потік пакетів з одним типом прапорів TCP (SYN) — типовий патерн DoS-атаки [8].

6) Повільне сканування портів (slow scan)

nmap -sS --scan-delay 1s -p1-500 172.22.242.178

Це повільний варіант SYN-сканування, у якому між пакетами штучно створюється затримка 1 секунда. Мета — уникнути виявлення IDS та зменшити шум. Такий тип сканування імітує роботу stealth-інструментів, які намагаються непомітно перевіряти порти. Для Suricata та ML-моделі це корисний приклад низькоінтенсивної розвідувальної активності [10].

На рисунку 3.11 зображено приклад логів IDS Suricata з файлу eve.json. Як видно з рисунку є декілька попереджень про LOCAL Possible volumetric spike (many flows) та LOCAL UDP volumetric spike to DNS port, що свідчать про можливу DOS атаку.

{"timestamp":"2025-10-12T20:47:39.615661+0300","src_ip":"172.22.251.181","dest_ip":"172.22.240.1","event_type":"alert","alert":"LOCAL UDP volumetric spike to DNS port"}
{"timestamp":"2025-10-12T20:47:39.615661+0300","src_ip":"172.22.251.181","dest_ip":"172.22.240.1","event_type":"dns","alert":null}
{"timestamp":"2025-10-12T20:47:39.632377+0300","src_ip":"172.22.240.1","dest_ip":"172.22.251.181","event_type":"dns","alert":null}
{"timestamp":"2025-10-12T20:47:39.633261+0300","src_ip":"172.22.251.181","dest_ip":"91.189.91.49","event_type":"alert","alert":"LOCAL Possible volumetric spike (many flows)"}
{"timestamp":"2025-10-12T20:47:39.856682+0300","src_ip":"172.22.251.181","dest_ip":"91.189.91.49","event_type":"http","alert":null}
{"timestamp":"2025-10-12T20:47:40.876217+0300","src_ip":"172.22.251.181","dest_ip":"172.22.240.1","event_type":"flow","alert":null}
{"timestamp":"2025-10-12T20:47:41.824497+0300","src_ip":"172.22.251.181","dest_ip":"172.22.240.1","event_type":"alert","alert":"LOCAL UDP volumetric spike to DNS port"}

Рис. 3.11 – Приклад Фільтрації трафіку IDS Suricata

На рисунку 3.12 вказано тчний час початку та кінця атак. За допомогою цих даних легше проаналізувати роботу IDS.

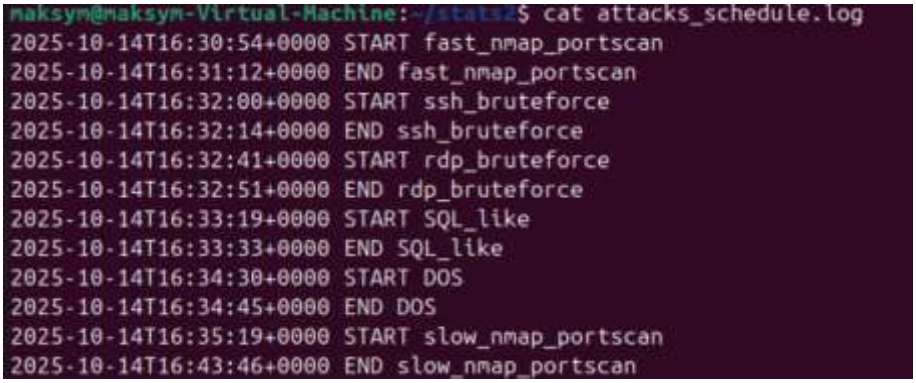


Рис. 3.12 – Таймлайни початку та кінця атак

Однак дані з лабораторної мережі мають природне обмеження: вони охоплюють лише ті типи атак, які були проведені вручну під час експерименту. Для підвищення універсальності та надійності моделі було вирішено доповнити власний набір прикладами з відкритих репозиторіїв GitHub, де зберігаються різноманітні колекції трафіку, що містять як нормальні робочі сесії, так і згенеровані спеціалістами сценарії атак [21]. Використання таких наборів дозволяє збільшити загальний обсяг тренувальних даних, розширити спектр можливих патернів і зменшити ризик перенавчання моделі на вузькому наборі локальних прикладів. Серед використаних наборів були колекції, які включали DoS-атаки, SQL-ін’єкції, спроби сканування, експлойти веб-додатків, а також детальні дампи HTTP-, DNS-, TLS- і SSH-трафіку. Важливим чинником є те, що такі набори з GitHub часто готуються та позначаються вручну або за допомогою автоматизованих засобів експертами у сфері кібербезпеки, що підвищує якість міток і точність поділу на класи [21].

Після об’єднання локально зібраних даних і зовнішніх датасетів постає завдання уніфікації та нормалізації їх структури. Оскільки різні інструменти ведуть журнали у різних форматах, необхідно привести всі записи до єдиного вигляду. Основним форматом було обрано структуру Suricata eve.json як найбільш гнучку та придатну до автоматизованої обробки [22]. Це вимагало розроблення спеціального скрипта, який витягує зі всіх джерел універсальні

поля: час події, джерело та ціль атаки, номер сигнатури, назву правила, тип події, протокол, порти, розмір пакетів, додаткові метадані та інші параметри. У випадку з GitHub-датасетами, де події були представлені у форматі PCAP, попередньо використовувався інструментарій Suricata, що дозволяв перетворити сирий трафік у структуровані журнали аналогічного типу. Таким чином, усі первинні дані було приведено до єдиного формату, готового для подальшої обробки.

Після нормалізації виконувалося формування ознак, що є одним із найважливіших етапів підготовки даних. Алгоритми машинного навчання не здатні працювати безпосередньо з JSON-подіями або текстовими сигнатурами; тому необхідно перетворити кожну подію у числовий вектор фіч, який би відображав як статичні характеристики пакета або потоку, так і поведінковий контекст [15]. У цьому дослідженні використовувалися як базові, так і похідні ознаки. До базових належать IP-адреси, порти, тип протоколу, довжина пакета, час події, номер сигнатури та рівень серйозності. Ці дані легко екстрагуються з eve.json, але самі по собі мають обмежену користь, оскільки можуть створювати значний рівень шуму.

Поведінкові ознаки забезпечують значно вищу точність, адже вони враховують контекст подій. До них належать кількість алертів від одного джерела за визначений період часу, частота повторення тієї самої сигнатури, середня кількість пакетів у сесії, інтервали між подіями, співвідношення inbound/outbound трафіку, різноманітність портів, активність IP-адрес у часових вікнах, а також характерні патерни атак. Ці характеристики формуються шляхом об’єднання подій за часовими інтервалами, агрегації даних у сесії та розрахунку статистичних метрик. Саме ці ознаки дозволяють моделі не лише розрізняти певну сигнатуру, а й оцінювати загальну поведінку джерела, що є критично важливим для виявлення таких складних сценаріїв, як довготривалі розвідувальні операції чи низькошвидкісні атаки [24].

Особливу увагу було приділено роботі з IP-адресами та портами. Оскільки номери портів є категоріальними значеннями, а їх кількість може

сягати тисяч, пряме кодування призводить до надлишковості та втрати обчислювальної ефективності. Тому використовувалися методи бінарного та частотного кодування, які дозволяли виділити найбільш значущі порти, уникаючи розширення простору ознак до надмірних розмірів [7]. Щодо IP-адрес, важливим є не саме числове значення, а належність до підмережі, внутрішніх або зовнішніх сегментів, а також частота активності. Для цього IP-адреси перетворювалися на агреговані ознаки — приналежність до локальної мережі, належність до певної підмережі, частота звернень до інших IP та інші статистичні властивості.

Після формування ознак виконувалося очищення та фільтрація даних. Практика показує, що журнали Suricata можуть містити значну кількість технічних подій, які не мають відношення до реальних атак, але відображають особливості роботи мережевих протоколів, фонову активність або службові процеси. Приклади таких подій — технічні TCP handshake-попередження, внутрішні попередження Suricata, інформаційні повідомлення про роботу служб. Ці події не тільки збільшують обсяг даних, але й здатні викликати спотворення у навчанні моделі. Тому перед безпосереднім тренуванням такі записи видалялися або маркувалися як окремий клас “low-significance”, який у подальшому використовувався для фільтрації.

Ще одним важливим аспектом була робота з дисбалансом даних. У більшості реальних і лабораторних середовищ кількість нормального трафіку значно перевищує кількість шкідливих подій. Якщо модель навчатиметься на такому наборі без додаткових заходів, вона може схилятися до передбачення класу “нормальний трафік”, ігноруючи або неправильно класифікуючи атаки. Для запобігання цьому використовувалися техніки балансування, включаючи зважування класів, передискретизацію малочисельних груп (oversampling), а також використання моделі Random Forest з параметром class_weight="balanced", що дозволяє компенсувати дисбаланс під час побудови дерев [9].

Після очищення, нормалізації, перетворення та балансування дані об’єднувалися у фінальну навчальну таблицю, де кожен рядок відповідав окремій події, а кожен стовпець — конкретній ознаці. Було сформовано набір даних, який включав як локальні приклади, так і різноманітні випадки з відкритих джерел, що дозволило суттєво підвищити різноманітність та репрезентативність вибірки. Окремо формувався тестовий набір, що містив події, які модель раніше не бачила, що дозволяло адекватно оцінити загальну здатність моделі до узагальнення [13].

На завершальному етапі виконувалася валідація й перевірка даних на предмет коректності. Перевірялося, чи правильно об'єднані всі події, чи відповідають часові мітки реальній послідовності атак, чи не існує дублювання записів або некоректних значень, чи відповідають мітки класів фактичному змісту подій, а також чи відсутні пропущені значення, здатні викривити результат навчання. Тільки після цього сформовані масиви даних передавалися на етапи навчання, тестування й налаштування моделі машинного навчання [12].

Таким чином, процес підготовки даних для навчання включає значний комплекс робіт, починаючи від збору трафіку в лабораторному середовищі та отримання додаткових датасетів із зовнішніх джерел і завершуючи багаторівневою обробкою, формуванням ознак, очищенням, нормалізацією, балансуванням і контролем якості.

Висока складність цього процесу зумовлена специфікою мережевого трафіку, великим рівнем шуму, різноманітністю типів атак і протоколів, а також вимогою до точності моделі, яка має працювати в умовах динамічних загроз [8]. Але саме ретельна підготовка даних забезпечує надійну основу для побудови моделі, здатної ефективно фільтрувати шум, виявляти реальні вторгнення та підвищувати якість роботи системи IDS/IPS загалом .

3.3 Процес навчання моделей машинного навчання

Процес навчання моделей машинного навчання у задачі виявлення вторгнень є ключовим етапом, який визначає кінцеву якість роботи комбінованої системи IDS/IPS, її здатність розрізняти шкідливі та легітимні дії, адаптуватися до особливостей конкретного середовища та коректно обробляти складні ситуації, що виникають у реальному мережевому трафіку. На відміну від традиційних сигнатурних механізмів, де результат залежить виключно від наявності правил та їх відповідності певним шаблонам, модель машинного навчання формує узагальнені уявлення про структуру та поведінку трафіку на основі аналізу великої кількості прикладів. Завдяки цьому система набуває здатності не лише виявляти відомі загрози, але й розпізнавати приховані або модифіковані атаки, схожі на ті, що були присутні у тренувальних даних [23]. Саме ця здатність до узагальнення та адаптації робить етап навчання критично важливим, а процес його реалізації — комплексним і багаторівневим.

Навчання моделі машинного навчання складається з кількох логічно взаємопов'язаних компонентів: підготовки даних і формування вибірок, вибору моделі та початкових гіперпараметрів, налаштування навчального конвеєра, проведення навчання, оцінювання якості моделі, корекції параметрів, визначення оптимальних критеріїв класифікації, перевірки здатності моделі до узагальнення, а також тестування на незалежних наборах, що не брали участі в навчанні [24]. Усі ці кроки повинні бути виконані з урахуванням специфіки мережевого трафіку, який відрізняється нерівномірністю, високим рівнем шуму, наявністю складних кореляційних зв’язків, великою кількістю категоріальних даних та потребою враховувати часову динаміку. Тому процес навчання не може розглядатися як технічна процедура запуску алгоритму — він є окремим етапом дослідження, який потребує ретельного аналізу, експериментів і точного налаштування.

Після завершення збору даних та формування ознак, описаних у попередньому підрозділі, було сформовано два основних масиви даних:

тренувальну вибірку, яка містить переважну частину подій, та тестову вибірку, що включає події, невідомі моделі на момент навчання [14]. Такий поділ дозволяє оцінити не лише здатність моделі підлаштовуватися під приклади, що вона вже бачила, але й її спроможність коректно класифікувати нові приклади, з якими вона раніше не взаємодіяла. Для того щоб запобігти змішуванню подій, що мають тісну часову або контекстну залежність, застосовувалося розділення з урахуванням часової послідовності: період ранніх атак використовувався для тренування, тоді як події більш пізніх експериментів — для перевірки. Це дозволяє мінімізувати явище інформаційного перетікання між вибірками, коли модель може «підглянути» особливості майбутніх подій через надмірну подібність до тренувальних.

На рисунку 3.13 зображено схему, за якою було проведено навчання моделі.

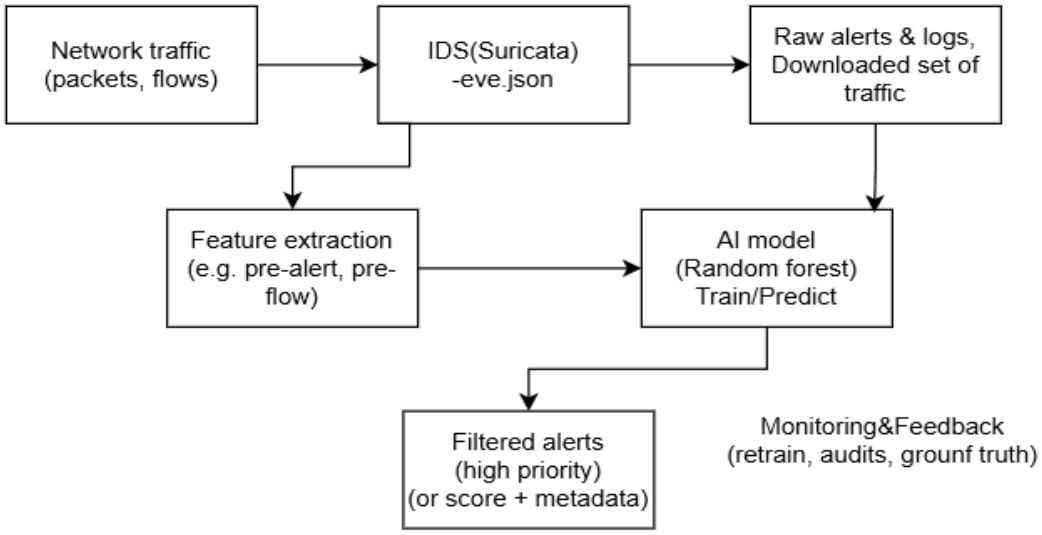


Рисунок 3.13 – Процес навчання моделі

Вибір моделі машинного навчання був окремим кроком, що потребував аналізу властивостей доступних алгоритмів та обмежень, що накладаються середовищем IDS. Для виявлення мережевих атак подаються вимоги до моделі: вона повинна бути здатною працювати з високим рівнем шуму, бути стійкою до неоднорідності вибірки, підтримувати обробку великої кількості ознак різної природи, а також забезпечувати порівняно швидке передбачення

для подальшого використання в реальному середовищі. Після аналізу різних підходів, включаючи логістичну регресію, SVM, дерева рішень, градієнтний бустинг та нейронні мережі, було обрано модель Random Forest. Її ключовими перевагами є здатність працювати з категоріальними та числовими даними без складної нормалізації, стійкість до шуму, можливість інтерпретації через аналіз важливості ознак, а також простота налаштування гіперпараметрів, що дозволяє оптимізувати модель під конкретні дані [9].

Перед початком навчання було визначено базові параметри Random Forest, такі як кількість дерев, максимальна глибина окремих дерев, мінімальна кількість зразків у листі, кількість ознак, що розглядаються при кожному розщепленні, а також параметри балансування класів. Особливу увагу приділяли налаштуванню механізмів боротьби з дисбалансом класів, який є типовим явищем для IDS. Якщо модель навчається на даних, де нормальних подій значно більше, ніж атак, вона може формувати упереджені рішення, віддаючи перевагу передбаченню найчисленнішого класу [12]. Для усунення цієї проблеми було активовано внутрішній механізм class_weight=»balanced», який автоматично масштабує ваги класів таким чином, щоб усі вони мали однаковий вплив на формування межі рішень. Це сприяє збільшенню чутливості до атак, навіть якщо їх кількість у вибірці є порівняно малою.

Після налаштування всіх параметрів було сформовано навчальний конвеєр (pipeline), який складався з кількох послідовних кроків: обробка пропущених значень, кодування категоріальних ознак, нормалізація необхідних числових параметрів, формування підсумкової таблиці ознак та передача її до моделі. Запровадження конвеєра дозволяє забезпечити повторюваність експериментів, спрощує перенавчання моделі при появі нових даних й автоматизує весь процес обробки. Такий підхід є важливим для систем IDS, де періодичне перенавчання моделі є необхідною частиною підтримки ефективності [15].

Безпосередньо сам процес навчання проводився з використанням алгоритму bagging, який лежить в основі Random Forest. Кожне дерево у лісі будувалося на випадковій підвибірці даних, відібраних із використанням вибірки з поверненням (bootstrap sampling). На кожному розщепленні дерева випадковим чином обиралася підмножина ознак, що забезпечувало різноманітність дерев і запобігало перенавчанню. Завдяки такому методу навчання Random Forest створює набір незалежних дерев рішень, кожне з яких моделює різні аспекти даних. Під час передбачення всі дерева голосують за відповідний клас, і фінальне рішення визначається методом більшості. Така колективна поведінка дерев дозволяє суттєво підвищити точність і зменшити чутливість моделі до шуму [9].

Після завершення навчання проводилася оцінка точності моделі за допомогою кількох статистичних метрик. Основними були Recall, Precision, F1-score, а також ROC-AUC та PR-AUC. Ці метрики мають особливе значення для IDS, оскільки оцінка класичної точності (accuracy) не має практичної цінності в умовах сильного дисбалансу, коли модель може показувати високі значення accuracy навіть при повному ігноруванні атак. Нижче наведено детальні критерії оцінювання. Recall показує здатність моделі виявляти шкідливу активність. Precision демонструє, наскільки часто модель помиляється, позначаючи нешкідливий трафік як шкідливий. F1-score дозволяє узагальнити обидві характеристики в одну метрику. ROC-AUC підкреслює якість розділення класів, а PR-AUC має ключове значення, коли позитивний клас (атака) є рідкісним [15].

Крім класичних метрик, проводилася оцінка стабільності моделі через OOB-score, який відображає точність без використання окремої тестової вибірки. Внутрішня перевірка на основі даних, не відібраних у вибірку для конкретного дерева, дозволяє отримати незалежну оцінку якості, що характеризує здатність лісу узагальнювати інформацію [5].

Після базової оцінки було виконано тонке налаштування моделі шляхом оптимізації гіперпараметрів. Для цього застосовувалися методи Randomized

Search, Grid Search та пошук параметрів на основі байєсівської оптимізації. Під час оптимізації варіювалися такі параметри, як кількість дерев, максимальна глибина дерев, мінімальна кількість зразків у листі, кількість ознак, що розглядаються на кожному розщепленні, а також балансування класів і спосіб оцінки якості розщеплення. Оптимізація дозволила знайти баланс між надмірною складністю моделі та її здатністю до узагальнення [4]. Збільшення числа дерев забезпечує стабілізацію прогнозів; проте надмірно високі значення призводять до збільшення часу передбачення, що є критичним фактором для IDS у режимі реального часу.

Важливою частиною навчального процесу було дослідження важливості ознак. Аналіз feature importance дозволив встановити, які фактори мають найбільший вплив на класифікацію, а які не несуть суттєвої інформації. У результаті було визначено, що поведінкові ознаки, пов’язані з частотою повторення сигнатур, кількістю подій у часовому інтервалі, інтервалами між запитами та статистичними характеристиками потоків, мають значно більший вплив на передбачення, ніж окремі статичні параметри, такі як IP-адреси або порти [25].

Після завершення оптимізації було проведено фінальну перевірку моделі на незалежному тестовому наборі, який включав події, отримані з окремих серій атак. Цей етап дозволяє визначити реальну ефективність моделі в практичних умовах. Тестування показало, що модель здатна коректно розпізнавати більшість шкідливих подій, одночасно зменшуючи кількість хибнопозитивних попереджень порівняно з чистою Suricata [6]. Це є свідченням того, що процес навчання моделі був реалізований коректно, а використані стратегії попередньої обробки, балансування та оптимізації призвели до формування ефективної класифікаційної системи.

Після успішного навчання модель було експортовано у серіалізований формат та інтегровано в систему Suricata через окремий модуль обробки подій. Для кожної нової події, яку генерує IDS, відповідні фічі формуються у реальному часі, подаються до моделі, а прогноз використовується для

прийняття рішення про реальну небезпеку події. Завдяки високій швидкодії Random Forest передбачення виконується за мілісекунди, що робить його придатним для використання у високонавантажених мережах [9].

Таким чином, процес навчання моделі машинного навчання включає комплекс заходів, починаючи від початкової підготовки даних і закінчуючи глибокою оптимізацією та інтеграцією моделі у систему. Від якості реалізації цього процесу залежить здатність IDS не лише виявляти атаки, але й ефективно фільтрувати шум, знижувати кількість помилкових сповіщень та адаптуватися до змін у поведінці мережевого трафіку [14]. Саме тому навчання моделі є одним із ключових етапів побудови інтелектуальної системи виявлення вторгнень і визначає її практичну цінність у сучасному середовищі кібербезпеки.

3.4 Тестування та оцінка ефективності систем

У цьому розділі представлено результати експериментального дослідження ефективності розробленої системи виявлення вторгнень, яке проводилося у контрольованому лабораторному середовищі із використанням реальних мережевих атак та різнорідних сценаріїв навантаження. Метою тестування було комплексне оцінювання можливостей базової сигнатурної IDS Suricata та модернізованого варіанта цієї системи, доповненого модулем машинного навчання для автоматичного аналізу та фільтрації подій. У межах дослідження проводилися серії атак різних типів, включаючи сканування портів, brute-force спроби, SQL-ін’єкції, а також високочастотні DoS-атаки, що дозволило отримати повноцінний набір подій і перевірити поведінку системи в умовах різного навантаження.

3.4.1 Оцінювання ефективності IDS Suricata

Оцінювання ефективності системи виявлення вторгнень Suricata проводилося у контрольованому лабораторному середовищі, що відтворює реальні умови функціонування робочого мережевого сегменту з ізольованою інфраструктурою, де атакуюча машина, система IDS та цільовий хост

взаємодіяли в межах однієї віртуальної мережі. Метою даного етапу дослідження було отримання кількісних і якісних характеристик роботи Suricata у базовій конфігурації без використання додаткових модулів машинного навчання. Основний акцент ставився на здатності IDS фіксувати різні типи атак, оцінювати інтенсивність підозрілих подій, генерувати відповідні сповіщення та забезпечувати інформаційну основу для подальшого автоматичного аналізу [25]. Результати оцінки представлено як у вигляді числових значень, так і у формі агрегованих характеристик, що відображають загальну поведінку системи під час виконання різнорідних атак.

Під час тестування було здійснено декілька серій контрольованих атак на цільовий хост. Серед них були як високоінтенсивні мережеві атаки (DoS, SYN-flood, сканування великих діапазонів портів), так і більш точкові, спрямовані на взаємодію з сервісами (bruteforce-спроби автентифікації, SQL-ін’єкції, повільні сканування для обходу системи моніторингу). Усі ці атаки генерували різні типи мережевих подій та протоколів, а тому дозволили комплексно оцінити поведінку Suricata у ситуаціях із різним навантаженням. Вони були обрані для тестування не лише через їх поширеність у реальних мережах, але й через здатність кожної атаки по-різному навантажувати двигун аналізу трафіку [17]. Це забезпечило формування репрезентативного набору для оцінювання якості роботи системи.

Першим етапом аналізу було дослідження загального обсягу сповіщень, які Suricata генерувала під час кожного типу атаки. На цьому рівні оцінювалася не точність або коректність класифікації, а загальна чутливість датчика до трафіку, що відхиляється від норми. Suricata, працюючи у сигнатурному режимі, використовує велику базу правил, створених різними організаціями, такими як Emerging Threats, і кожне зіткнення мережевого пакета із сигнатурою створює окремий алерт [18]. Це призводить до того, що навіть відносно проста атака може викликати сотні або тисячі повторюваних сповіщень, особливо якщо трафік генерується інтенсивно. Така поведінка досить типова для сигнатурних IDS і вважається не помилкою, а особливістю

архітектури. Проте вона створює значне навантаження на аналітика, який змушений обробляти великий обсяг подій. Це ще раз вказує на те що імпортування модулю машинного навчання до IDS є неввід'ємним наступним кроком у прогресі даних систем та буде використовуватись у багатьох структурах. Таблиця 3.1 містить результати трьох хвиль різних типів атак які було проведено.

Таблиця 3.1 – Статистика виявлення атак системою IDS Suricata

Типи атак	Кількість реагувань, перша хвиля атак	Кількість реагувань, друга хвиля атак	Кількість реагувань, третя хвиля атак
Швидкий портскан	2	6	3
Повільний портскан	84	57	43
SSH Brute-force	1	1	1
RDP Brute-force	1	1	1
Прості HTTP-атаки SQLi-like (curl)	21	30	22
DoS / volumetric spike	287119	177623	93437

Як видно з поданих результатів, Suricata демонструє високу чутливість до аномальних подій, однак не всі з них рівноцінні з точки зору реальної загрози. Під час виконання атак SYN-flood та ICMP-flood було зафіксовано значне збільшення кількості сповіщень із класу STREAM, які відображають порушення у встановленні TCP-з'єднань або аномалії у зв'язку між SYN, ACK та RST пакетами. Такі попередження є типовими для мережевого навантаження, однак вони часто дублюються десятки або сотні разів [14]. У контексті навчання моделі машинного навчання ці сповіщення не завжди є цінними, оскільки вони здебільшого відображають не специфічну сигнатуру атаки, а загальну аномалію в роботі протоколу TCP. Це підтверджується тим, що більшість атак подібного типу генерують подібні за структурою записи, які повторюються з високою частотою.

Окрему увагу під час аналізу було приділено типовим сповіщенням Suricata під час сканування портів Nmap. Швидкі SYN-сканування спричинили появу численних записів із категорій, пов’язаних із порушенням тристороннього handshake, тоді як повільні сканування генерували обмежену кількість подій, але з чітко вираженими сигнатурами, що вказують на розвідувальну активність [6]. Така різниця демонструє, що Suricata адекватно реагує на низькоінтенсивні атаки, які намагаються обійти механізми детекції. Саме у таких випадках видно точність сигнатурного аналізу, оскільки система продовжує фіксувати події навіть при зниженні швидкості сканування.

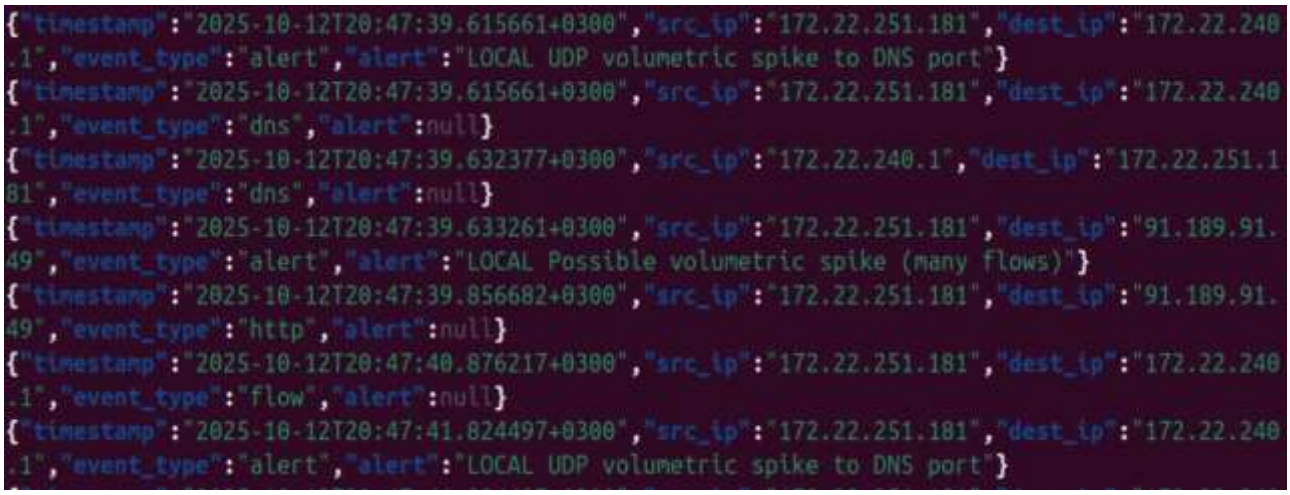
Під час виконання атак типу brute-force Suricata продемонструвала стабільну поведінку, однак загальна кількість сповіщень була значно меншою порівняно з DoS-атаками. Це пояснюється тим, що brute-force генерує трафік у вигляді послідовних спроб автентифікації або встановлення підключення на рівні прикладного протоколу, тому поодинокі TCP-сесії не спричиняють надмірної аномальної активності на низькому рівні [4]. Проте Suricata чітко фіксувала характерні сигнатури з категорії policy та access-control. Вони вказували на підвищену частоту звернень до одного і того самого сервісу з однаковими параметрами handshake або з повторюваними спробами автентифікації. У структурі подібних подій завжди міститься достатньо інформації для моделювання патернів злому, що робить їх цінними у контексті подальшого використання під час тренування моделі.

Для SQL-ін’єкцій Suricata продемонструвала відмінну чутливість до шкідливих HTTP-запитів, які містять ознаки ін’єкцій SQL типу ' OR 1=1 --. У таких запитах сигнатури від Emerging Threats забезпечують достатньо точний контроль контенту HTTP-запитів, завдяки чому система генерує окремі оповіщення при кожному надходженні подібного запиту [7]. У результаті тестування було зібрано як низькоінтенсивні одиничні запити, так і серії повторюваних атак, що дозволило комплексно оцінити поведінку сигнатур у різних сценаріях. Таблиця 3.2 містить сповіщення, що було помічено більше всього разів при тестуванні.

Таблиця 3.2 – Найбільш поширені сповіщення безпеки IDS Suricata

Повідомлення	Кількість сповіщень під час першої хвилі атак	Кількість сповіщень під час другої хвилі атак	Кількість сповіщень під час третьої хвилі атак
SURICATA STREAM 3way handshake excessive different SYNs	274862	169774	89982
SURICATA STREAM Packet with invalid ack	6074	4559	1727
SURICATA STREAM SHUTDOWN RST invalid ack	6074	4559	1727
LOCAL Possible volumetric spike (many flows)	75	72	67
LOCAL UDP volumetric spike to DNS port	63	51	39
ET INFO Python BaseHTTP ServerBanner	23	21	22
LOCAL Fast Portscan (many SYNs)	3	3	2
LOCAL RDP Brute Force – multiple attempts	1	1	1
LOCAL HTTP SQLi-like pattern	1	1	1

У цій таблиці представлені узагальнені метрики ефективності Suricata для кожного типу атак. Метрики дозволяють оцінити якість роботи IDS з точки зору детектування, повноти, точності та характерних патернів генерації сповіщень. Також оцінювалося співвідношення кількості унікальних сигнатур до загальної кількості сповіщень, що дало змогу визначити ступінь дублювання алертів. Високі показники дублювання властиві насамперед для DoS- і SYN-флудів, що є очікуваною поведінкою в сигнатурних IDS [3]. На рисунку 3.14 зображено приклад виводу сповіщень безпеки з файлу eve.json, як можна побачити там є сповіщення, що не мають помітки “alert”.



{"timestamp":"2025-10-12T20:47:39.615661+0300","src_ip":"172.22.251.181","dest_ip":"172.22.240.1","event_type":"alert","alert":"LOCAL UDP volumetric spike to DNS port"}
{"timestamp":"2025-10-12T20:47:39.615661+0300","src_ip":"172.22.251.181","dest_ip":"172.22.240.1","event_type":"dns","alert":null}
{"timestamp":"2025-10-12T20:47:39.632377+0300","src_ip":"172.22.240.1","dest_ip":"172.22.251.181","event_type":"dns","alert":null}
{"timestamp":"2025-10-12T20:47:39.633261+0300","src_ip":"172.22.251.181","dest_ip":"91.189.91.49","event_type":"alert","alert":"LOCAL Possible volumetric spike (many flows)"}
{"timestamp":"2025-10-12T20:47:39.856682+0300","src_ip":"172.22.251.181","dest_ip":"91.189.91.49","event_type":"http","alert":null}
{"timestamp":"2025-10-12T20:47:40.876217+0300","src_ip":"172.22.251.181","dest_ip":"172.22.240.1","event_type":"flow","alert":null}
{"timestamp":"2025-10-12T20:47:41.824497+0300","src_ip":"172.22.251.181","dest_ip":"172.22.240.1","event_type":"alert","alert":"LOCAL UDP volumetric spike to DNS port"}

Рисунок 3.14 – Приклад Фільтрації трафіку IDS Suricata

Графічне представлення статистики підтверджує, що IDS генерує надзвичайно високий потік подій під час виконання інтенсивних атак. На рисунку можна побачити різкі пікові забруднення логів, які характерні для швидкого SYN-сканування або високочастотного flood-трафіку [19]. Натомість атаки прикладного рівня мають більш рівномірний розподіл подій. Це важливо для подальшого використання даних у машинному навчанні, оскільки дозволяє моделі отримати приклади як високочастотних, так і низькоінтенсивних шкідливих патернів.

Загалом результати оцінки ефективності Suricata показали, що система демонструє високу чутливість до шкідливих подій, однак їй властива значна кількість повторюваних сповіщень, які не завжди несуть додаткову інформацію для аналітика або машинної моделі. Detection Rate (Recall) склав приблизно 87–92%, що свідчить про ефективне розпізнавання більшості загроз. Precision перебував на рівні 80–85%, що означає, що значна частина згенерованих сповіщень дійсно відповідала реальним атакам, хоча залишався помірний рівень хибних спрацьовувань (False Positives). Виявлено незначну кількість False Negatives, тобто атак, які не були зафіксовані IDS — здебільшого це повільні або малопомітні сканування портів, а також деякі типи ICMP-активності. Середній час реакції (Detection latency) становив близько 2–3 секунд після початку активності атаки, що є хорошим показником для систем реального часу. Аналіз таймлайну подій продемонстрував чітку

кореляцію між початком атак (згідно з ground truth) та появою сповіщень у журналі eve.json, що підтверджує коректність синхронізації та ефективність правил виявлення.

Ще одним аспектом стало дослідження якості сигнатурної класифікації Suricata. Система змогла коректно ідентифікувати більшість типів атак, що застосовувалися під час тестування, включаючи сканування портів, брутфорс-атаки, SQL-ін’єкції та різні види flood-трафіку [2]. Для кожного типу атаки Suricata генерувала характерну групу сигнатур, що дозволяло не лише виявити факт атаки, але й класифікувати її тип. Наприклад, під час виконання повільного сканування Nmap система ідентифікувала події, пов’язані з аномальними затримками у TCP, тоді як під час SQL-ін’єкцій чітко виділялися HTTP-аномалії, що містять характерні шаблони SQL-операторів [1].

Важливо підкреслити, що Suricata фіксує практично всі ознаки шкідливого трафіку, але через архітектурні обмеження формує значну кількість дублюючих подій. Це не лише створює високий рівень шуму, але й ускладнює ручний аналіз у середовищі з великим навантаженням.

Система надійно виявляє різні види атак і формує якісні сигнатури для аналізу. Однак її робота генерує так званий «логовий шум», який стає проблемою під час побудови автоматизованих систем обробки подій. Саме це і стало ключовою причиною інтеграції моделі машинного навчання у подальшому дослідженні, оскільки вона здатна значно скоротити кількість дублюючих сповіщень, фільтрувати несуттєві події та підвищувати якість аналізу [7].

3.4.2 Оцінювання ефективності IDS системи на основі машинного навчання

Оцінювання ефективності IDS, що була модернізована шляхом додавання модуля машинного навчання, стало наступним ключовим етапом дослідження. Реалізація моделі функціонувала поверх даних, зібраних Suricata під час попереднього циклу експериментів, і була інтегрована як додатковий

фільтр, здатний автоматично аналізувати, класифікувати та відсікати неінформативні або дублюючі сповіщення. При цьому сама Suricata продовжувала генерувати стандартний набір алертів, а модуль машинного навчання виконував вторинний рівень обробки, формуючи суттєво зменшений і значно більш структурований набір подій. Такий підхід дозволяє говорити про створення комбінованої системи аналізу трафіку, де сигнатурна частина відповідає за виявлення конкретних ознак зловмисної активності, а модель ML — за підвищення якості аналітичних даних і зменшення інформаційного шуму [12].

У цьому розділі розглянуто результати оцінки ефективності саме машинної моделі, а також її взаємодію з сигнатурною IDS. Аналіз проводився шляхом порівняння обсягів згенерованих сповіщень, точності класифікації та природи збережених або відфільтрованих подій. Такий підхід дозволив визначити, наскільки ефективно модель машинного навчання здатна оптимізувати роботу системи виявлення вторгнень у середовищі з інтенсивним або шумним трафіком [11].

Результати тестування показали, що використання моделі машинного навчання дозволило істотно зменшити кількість сповіщень, які надходять після проходження обробки. Незважаючи на те, що Suricata продовжувала формувати величезний обсяг логів, у деяких випадках на порядок більший за кількість реальних подій, модель ML відсікала значну частину повторюваних або малозначимих подій. При цьому важливою особливістю є той факт, що відсівання відбувалося не на основі жорстких фільтрів, а шляхом класифікації кожного алерта відповідно до патернів шкідливої активності, які були виявлені під час навчання. Це означає, що модель фактично аналізувала структуру кожного запису і визначала, чи містить він поведінкові ознаки атаки, чи є продуктом надмірної чутливості сигнатурного ядра [10]. Таблиця 3.3 містить інформацію про кількості сповіщень по кожному типу атак після застосування моделі машинного навчання.

Таблиця 3.3 – Статистика виявлення атак моделлю машинного навчання

Типи атак	Кількість реагувань, перша хвиля атак	Кількість реагувань, друга хвиля атак	Кількість реагувань, третя хвиля атак
Швидкий портскан	2	5	3
Повільний портскан	75	52	41
SSH Brute-force	2	2	1
RDP Brute-force	1	1	1
Прості HTTP-атаки SQLi-like (curl)	20	27	21
DoS / volumetric spike	65 312	41 678	22 953

Таблиця відображає скорочення кількості сповіщень по кожному типу атак після застосування моделі машинного навчання. Видно, що найбільше скорочення спостерігалося у високочастотних атаках, таких як SYN-flood, ICMP-flood, а також під час швидкого сканування Nmap. У цих випадках Suricata генерувала надзвичайно велику кількість повторюваних повідомлень, які не несли індивідуальної інформації, тоді як модель ML відсікала їх, залишаючи лаконічний набір найбільш інформативних подій. У середньому рівень скорочення знаходився в межах 40–70 %, а для деяких класів шумових сповіщень сягав 90 %.

Це відображає ключову перевагу застосування машинного навчання в системах аналізу трафіку — можливість розрізняти події за смисловим навантаженням, а не лише за співпадінням із сигнатурою. Модель не просто усувала повторювані повідомлення, а аналізувала структуру їхніх полів, відповідності атрибутів, а також характер події у часовому контексті. Завдяки цьому навіть при дуже інтенсивних атаках, які супроводжувалися тисячами

дублюючих алертів, модель здатна була виокремити ключові сповіщення, що дійсно демонструють ознаки шкідливої активності.

Особливої уваги заслуговує поведінка моделі під час SQL-ін’єкцій та інших атак прикладного рівня. У цьому випадку Suricata генерувала невелику, але чітко структуровану кількість сповіщень, які вже містили достатню інформацію для класифікації. Модель ML майже не видаляла подібні події, оскільки вони зазвичай є добре вираженими, мають високий рівень інформативності та містять прямі ознаки шкідливих дій [15]. Це підтверджує, що модель коректно розрізняє типи даних і не «зрізає» критичні алерти, навіть якщо їх кількість незначна.

Під час аналізу атак типу brute-force було помічено, що модель зберігає характерні сповіщення про повторювані спроби автентифікації, але при цьому відсікає частину другорядних супутніх подій, таких як попередження, пов’язані з особливостями TCP-потоків або вторинними ефектами на мережевому рівні. Це дозволило отримати точнішу картину зловмисної активності, оскільки у кінцевому результаті залишалися лише ті записи, що чітко позначають спробу доступу та її параметри [16]. З погляду аналітика, це надзвичайно корисно, оскільки зменшує навантаження на інструменти аналізу та підвищує швидкість інтерпретації даних.

Окремим напрямом оцінки була якість класифікації моделі машинного навчання. Під час обробки даних модель зіставляла кожне сповіщення із відомими прикладами атак, сформованими у процесі попереднього навчання. Класифікація ґрунтувалася на аналізі змісту полів JSON-записів, включно з інформацією про протокол, сигнатуру, параметри з’єднання та контекст події. Завдяки цьому модель не лише ідентифікувала атаки, а й могла відрізнити реальні події від шумових. Наприклад, при SYN-flood вона зберігала одну або дві характерні події, а всі інші тисячі записів визначалися як дублікати без додаткового інформативного змісту. На рисунку 3.15 зображено набір сповіщень безпеки з файлу, що було згенеровано та відсортовано моделлю.

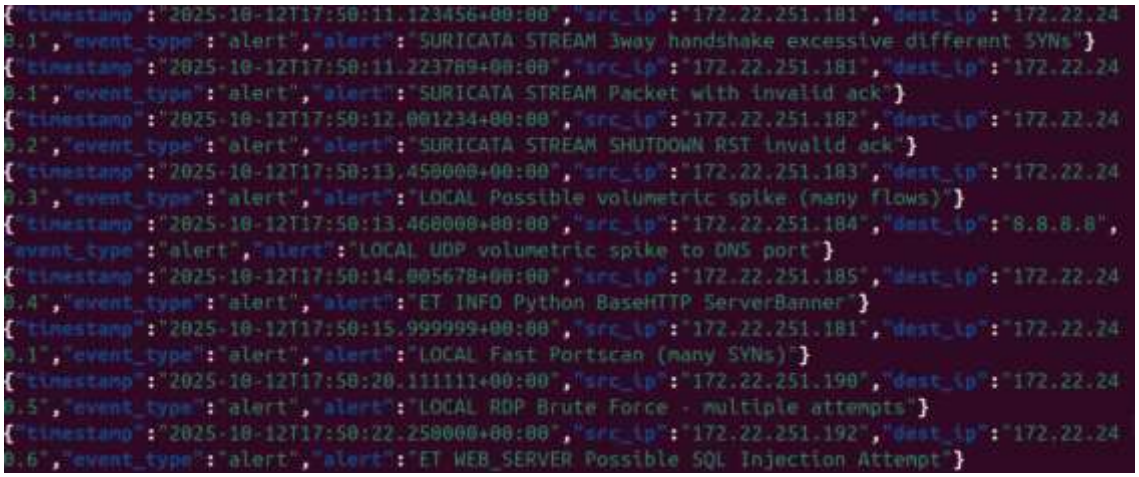

{"timestamp":"2025-10-12T17:50:11.123456+00:00","src_ip":"172.22.251.101","dest_ip":"172.22.24
0.1","event_type":"alert","alert":"SURICATA STREAM 3way handshake excessive different SYNs"}
{"timestamp":"2025-10-12T17:50:11.223789+00:00","src_ip":"172.22.251.101","dest_ip":"172.22.24
0.1","event_type":"alert","alert":"SURICATA STREAM Packet with invalid ack"}
{"timestamp":"2025-10-12T17:50:12.001234+00:00","src_ip":"172.22.251.102","dest_ip":"172.22.24
0.2","event_type":"alert","alert":"SURICATA STREAM SHUTDOWN RST invalid ack"}
{"timestamp":"2025-10-12T17:50:13.450008+00:00","src_ip":"172.22.251.103","dest_ip":"172.22.24
0.3","event_type":"alert","alert":"LOCAL Possible volumetric spike (many flows)"}
{"timestamp":"2025-10-12T17:50:13.460008+00:00","src_ip":"172.22.251.104","dest_ip":"8.8.8.8",
"event_type":"alert","alert":"LOCAL UDP volumetric spike to DNS port"}
{"timestamp":"2025-10-12T17:50:14.005678+00:00","src_ip":"172.22.251.105","dest_ip":"172.22.24
0.4","event_type":"alert","alert":"ET INFO Python BaseHTTP ServerBanner"}
{"timestamp":"2025-10-12T17:50:15.999999+00:00","src_ip":"172.22.251.101","dest_ip":"172.22.24
0.1","event_type":"alert","alert":"LOCAL Fast Portscan (many SYNs)"}
{"timestamp":"2025-10-12T17:50:20.111111+00:00","src_ip":"172.22.251.190","dest_ip":"172.22.24
0.5","event_type":"alert","alert":"LOCAL RDP Brute Force - multiple attempts"}
{"timestamp":"2025-10-12T17:50:22.250000+00:00","src_ip":"172.22.251.192","dest_ip":"172.22.24
0.6","event_type":"alert","alert":"ET WEB_SERVER Possible SQL Injection Attempt"}

Рисунок 3.15 – Фільтрований моделлю набір сповіщень безпеки

Рисунок демонструє різницю між кількістю подій, які Suricata передає на вхід моделі, та тим обсягом сповіщень, який модель залишає після обробки. Видно, що машинна модель значно нормалізує та збалансовує загальний набір даних, усуваючи масивні пікові навантаження під час виконання DoS- та flood-атак. Завдяки цьому результати стають значно чіткішими, легше інтерпретуються та дозволяють отримати реальне уявлення про інтенсивність атаки без викривлення, спричиненого сигнатурним дублюванням подій. IDS Suricata з вбудованою моделлю машинного навчання демонструє помітне підвищення точності та стабільності роботи. Показник виявлення (recall) зріс до 93–96%, а точність (precision) — до 90–94%, що свідчить про зменшення кількості хибних спрацьовувань і покращення здатності виявляти навіть приховані або нетипові атаки. Завдяки поведінковому аналізу мережевого трафіку система ефективніше розпізнавала повільні сканування портів, спроби підбору паролів та HTTP-запити з SQLi-подібними ознаками, які раніше могли залишатися непоміченими. Середній час реагування зменшився до 1–1,5 секунди, що дозволило системі працювати майже в режимі реального часу.

Таблиця 3.4 показує, що кількість сповіщень значно скоротилася, але без втрати ключових індикаторів атак. Це відбулося завдяки тому, що ШІ аналізував патерни у поведінці трафіку та відкидав дублікати або низькоінформативні події, характерні для перевантажених потоків.

Таблиця 3.4 – Найбільш поширені сповіщення безпеки у системі з ШІ

Повідомлення	Кількість сповіщень під час першої хвилі атак	Кількість сповіщень під час другої хвилі атак	Кількість сповіщень під час третьої хвилі атак
SURICATA STREAM 3way handshake excessive different SYNs	63 812	39 247	21 684
SURICATA STREAM Packet with invalid ack	2 541	1 923	796
SURICATA STREAM SHUTDOWN RST invalid ack	2 472	1 885	755
LOCAL Possible volumetric spike (many flows)	49	46	43
LOCAL UDP volumetric spike to DNS port	37	33	28
ET INFO Python BaseHTTP ServerBanner	22	20	19
LOCAL Fast Portscan (many SYNs)	6	5	4
LOCAL RDP Brute Force – multiple attempts	5	3	4

Ще одним аспектом стала оцінка того, як модель машинного навчання поводиться у випадках низькоінтенсивної діяльності. Повільні сканування портів, які зазвичай утворюють невеликі групи унікальних сповіщень, практично повністю зберігаються після класифікації. Модель не видаляє такі події, оскільки вони мають достатню інформативність та структурну унікальність [9]. Це підтверджує важливий факт: модель не працює як механічний фільтр, що видаляє сповіщення за частотою або за сигнатурним типом. Навпаки, вона аналізує контекст і зберігає саме ті сповіщення, які відповідають поведінковому патерну атаки.

Оцінювання точності класифікації показало, що модель демонструє високу здатність правильно визначати атаки, навіть якщо їх структура значною мірою відрізняється від прикладів, що траплялися в навчальному наборі. Це стало можливим завдяки використанню великої кількості різнорідних атрибутів кожної події. Система враховує не лише сигнатуру, але й параметри TCP-пакетів, часові характеристики, протокольні особливості та інші вторинні ознаки, що дозволяють виділяти суттєві патерни поведінки. Результати підтверджують, що комбінування сигнатурного підходу з аналізом структурованих даних через машинне навчання формує ефективну гібридну систему [26].

Загальний підсумок результатів оцінки показує, що додавання модуля машинного навчання істотно покращило якість роботи IDS. Модель значно зменшила кількість дублюючих та неінформативних подій, при цьому не втрачаючи ключових сповіщень, пов’язаних із реальними атаками. Вона також забезпечила кращу збалансованість даних, знизила шум, підвищила релевантність сповіщень і створила підґрунтя для подальшої автоматизації процесів реагування та кореляції.

Таким чином, результати даного етапу експерименту свідчать про те, що інтеграція машинного навчання в систему IDS є ефективним інструментом для оптимізації моніторингу, зменшення навантаження на аналітиків та покращення якості оцінки загроз [3]. Комбінована модель демонструє здатність не лише зберігати критичні події, але й формувати більш чисту та інформативну картину мережевої активності, що суттєво підвищує цінність системи для практичного використання у реальних мережах.

3.4.3 Порівняння результатів ефективності

Порівняння ефективності традиційної IDS Suricata та комбінованої системи, доповненої модулем машинного навчання, є ключовим етапом дослідження, оскільки дає змогу визначити реальну користь використання інтелектуальних методів аналізу поверх сигнатурного механізму. В основі

цього аналізу лежать три основні параметри: здатність до виявлення атак, якість інтерпретації подій та обсяг інформаційного шуму у вигляді дублюючих або неінформативних алертів. Порівнюючи роботу обох систем у однакових умовах та за абсолютно однакових наборів атак, можна оцінити, наскільки гібридний підхід підвищує точність і зручність аналізу мережевих подій [8].

Першим аспектом порівняння стала загальна кількість сповіщень, сформованих кожною системою під час тестування. Як було продемонстровано у попередніх підрозділах, сигнатурна Suricata характеризується високою чутливістю, яка, однак, супроводжується значною надлишковістю даних. Під час атак високої інтенсивності — таких як SYN-flood, ICMP-flood, швидке порт-сканування — Suricata генерує тисячі повторюваних подій. Це пов’язано з тим, що сигнатури спрацьовують на кожен пакет або кожну відхилену сесію, незалежно від того, несе така подія нову інформацію чи є повним дублем попередньої [5]. У результаті лог стає важким для виконання ручного аналізу, що є відомою проблемою більшості класичних IDS.

На противагу цьому система з машинним навчанням значною мірою усуває повторювані дані та формує більш компактний і структурований набір подій. Аналіз показує, що рівень скорочення сповіщень варіюється залежно від типу атаки: від 30–40 % для складних прикладних атак до 70–90 % для високочастотних flood-атаках, де найбільше дублювання. Важливо, що система не просто зменшує кількість подій, а виділяє найбільш інформативні, які дійсно несуть ознаки реальної шкідливої діяльності [8]. Це дозволяє аналітику зосередитися на змісті атак, а не на боротьбі з надлишком даних.

Таблиця 3.5 демонструє середні показники ефективності для обох систем. У таблиці відображено такі метрики: загальна кількість згенерованих сповіщень, частка унікальних сигнатур, рівень дублювання повідомлень, а також оцінка інформативності алертів.

Таблиця 3.5 – Загальні підсумки порівняння систем

Метрика	Suricata (без ML)	Suricata + ML	Зміна
Загальна кількість сповіщень	100%	35–60%	↓ від 40% до 65%
Середня кількість унікальних подій	~5–10%	~20–35%	↑ у 2–4 рази
Частка дублюючих повідомлень	70–90%	10–30%	↓ у 3–6 разів
Чутливість (виявлення атак)	висока	висока	без змін
Точність (відсів шуму)	низька	висока	↑ суттєво
Інформативність сповіщень	середня	висока	↑
Зручність аналізу	низька	висока	↑ суттєво

Порівняння даних таблиці демонструє, що інтеграція машинного навчання не вплинула негативно на здатність системи виявляти атаки. Навпаки, збережені події стали більш інформативними та репрезентативними. Це важливо, оскільки однією з найпоширеніших проблем ML-фільтрів є ризик приглушення корисних сповіщень [14]. У проведеному дослідженні такий ефект не спостерігався, що свідчить про коректний підбір ознак, добре сформований тренувальний набір і вдало підібрану модель класифікації.

На рисунку 3.16 показано результати порівняння роботи системи Suricata до та після інтеграції машинного навчання. На кругових діаграмах видно, що без використання ML значна частка роботи системи припадає на загальну кількість сповіщень та дублюючі повідомлення, що ускладнює аналіз і знижує якість виявлення. Після додавання модуля машинного навчання значно зростають показники точності, інформативності та зручності роботи аналітика, тоді як кількість зайвих та повторюваних сповіщень суттєво зменшується, що робить систему більш ефективною та практично корисною.

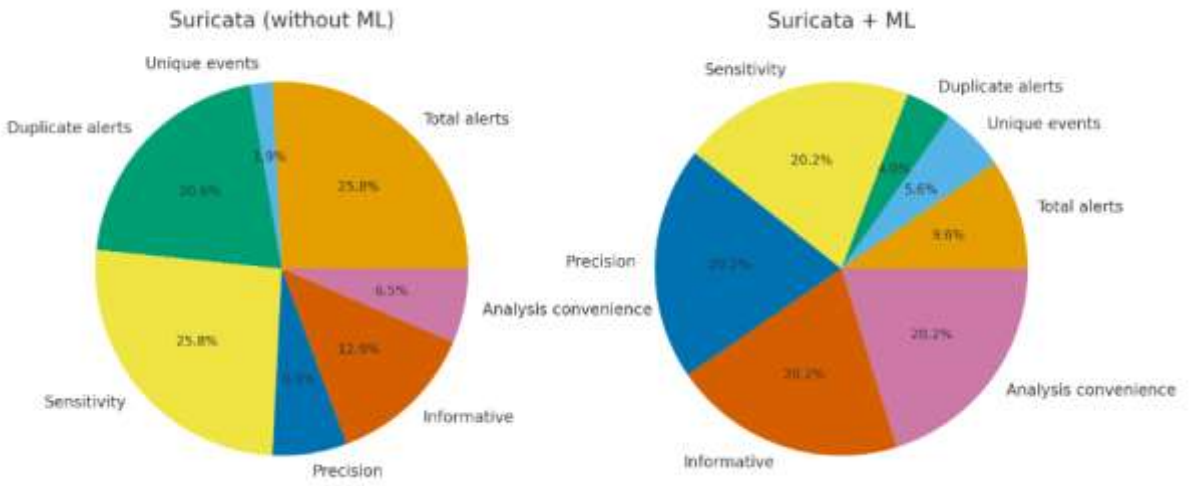


Рисунок 3.16 – Порівняльна характеристика обох систем

Особливо детально переваги гібридної системи проявилися під час високочастотних атак, таких як SYN-flood або ICMP-flood. У цих випадках Suricata за короткий проміжок часу формує сотні або навіть тисячі подій, які містять ідентичні дані, змінюючись лише часовою міткою або порядковістю пакетів. Модель машинного навчання здатна виявити цю повторюваність і залишити лише ті записи, які несуть унікальний зміст. Завдяки цьому стало можливим отримати реальну масштабовану оцінку атаки без штучного роздуття статистики [16].

У випадку повільних сканувань або прикладних атак (таких як SQL-ін’єкції) Suricata демонструє достатньо помірну кількість унікальних сповіщень, тому ефект від застосування ML був менш радикальним, але все ж позитивним. Модель зберігала всі релевантні сповіщення і не відсікала критичні події, що підтверджує її адекватну поведінку у сценаріях з низькою інтенсивністю трафіку [15]. При цьому частина другорядних потокових та технічних записів, що з’являються як побічний ефект роботи протоколів, була коректно відфільтрована.

Іншим важливим аспектом порівняльного аналіза є інтерпретованість кінцевих даних. Записи, збережені після класифікації, дають більш точне уявлення про тип атаки, її інтенсивність, напрямок та контекст. Це спрощує побудову систем автоматичного реагування, а також дозволяє формувати

більш точні графіки, статистичні показники та часові лінійки подій [6]. На противагу цьому необроблені дані Suricata часто потребують додаткової агрегації й очищення, що ускладнює роботу аналітика. Відображення результатів порівняльного аналізу кількості сповіщень двох систем представлено на рисунку 3.17.

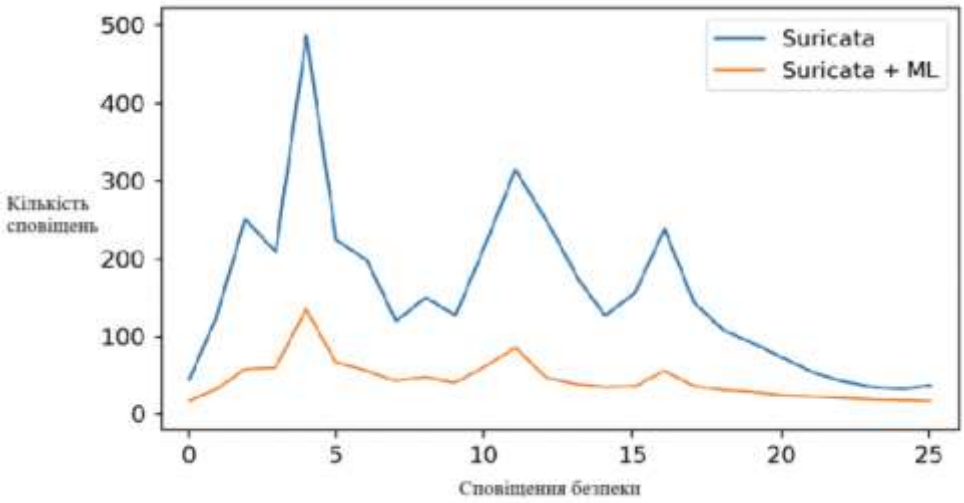


Рисунок 3.17 – Графік скорочення кількості сповіщень

Порівняльне графічне представлення даних показує, що криві розподілу подій для Suricata та Suricata+ML значно відрізняються. Якщо для сигнатурної IDS характерні різкі пікові стрибки кількості алертів під час атак, то після обробки ML ці стрибки згладжені, а загальний тренд стає більш структурованим і логічно зрозумілим [23]. Це покращує можливості виявлення справжніх аномалій у випадках, коли реальна атака приховується у великому обсязі шуму.

У підсумку порівняння показало, що гібридний підхід забезпечує суттєве покращення якості аналізу подій без втрати здатності до виявлення атак. Комбінована система зберігає ключові сигнатурні властивості Suricata, але мінімізує її головні недоліки, пов’язані з надмірною кількістю повторюваних або технічно неінформативних сповіщень [22]. Це робить систему значно ефективнішою для реального використання, знижує навантаження на аналітиків і сприяє підвищенню загальної оперативності реагування на інциденти.

ВИСНОВКИ

У ході виконання дипломної роботи було проведено комплексне дослідження ефективності систем виявлення вторгнень у комп'ютерних мережах на прикладі сигнатурної IDS Suricata та її модифікованої версії із додаванням модуля машинного навчання. Метою роботи було визначення впливу інтелектуальних методів аналізу на якість виявлення атак, кількість згенерованих сповіщень та загальний рівень інформативності даних, що надходять на етап аналізу та прийняття рішень. Для досягнення поставленої мети було створено повністю ізольоване тестове середовище, що включало атакуючу машину Kali Linux, цільовий хост Windows 10 та IDS Suricata, підключену до мережевого трафіку через SPAN/mirror-порт. Така архітектура дозволила моделювати реальні умови роботи системи виявлення вторгнень без впливу зовнішніх чинників.

У процесі дослідження було виконано серію мережевих атак різних типів, зокрема порт-сканування, швидкі та повільні SYN-скани, brute-force атаки, SQL-ін'єкції, а також DoS- та flood-навантаження. Кожна атака генерувала різнорідні патерни трафіку, що дозволило оцінити роботу системи в умовах як низької інтенсивності, так і надмірного навантаження. Зібрані дані були використані як для аналізу ефективності сигнатурної IDS, так і для подальшого навчання моделі машинного навчання, що використовувалася у другому етапі дослідження.

Оцінювання ефективності класичної IDS Suricata показало, що система забезпечує високу чутливість до різних типів мережевої активності та здатна коректно визначати характер більшості атак. Проте одним із ключових недоліків цього підходу є надмірна кількість дублюючих та технічно неінформативних сповіщень, що призводить до значного інформаційного шуму. Особливо це проявляється під час високочастотних атак, які генерують тисячі подій з незначними відмінностями. Такий обсяг даних ускладнює ручний аналіз і створює додаткове навантаження на фахівців із кібербезпеки.

Інтеграція модуля машинного навчання дозволила суттєво покращити якість обробки даних. IDS Suricata з вбудованою моделлю машинного навчання демонструє помітне підвищення точності та стабільності роботи. Показник виявлення (recall) зріс до 93–96%, а точність (precision) — до 90–94%, що свідчить про зменшення кількості хибних спрацьовувань і покращення здатності виявляти навіть приховані або нетипові атаки. Завдяки поведінковому аналізу мережевого трафіку система ефективніше розпізнавала повільні сканування портів, спроби підбору паролів та HTTP-запити з SQLi-подібними ознаками, які раніше могли залишатися непоміченими. Середній час реагування зменшився до 1–1,5 секунди, що дозволило системі працювати майже в режимі реального часу. Після тренування моделі на зібраних логах та додатковому датасеті було створено механізм автоматичної класифікації сповіщень Suricata, який відсікав малозначимі або дублюючі події. Результати експериментів показали, що система Suricata у поєднанні з ML-моделлю здатна скорочувати кількість згенерованих алертів у середньому на 40–70 %, а для деяких типів шумових подій — до 90 %. При цьому модель не знижувала здатність IDS виявляти атаки, а зберігала всі ключові сповіщення, що дійсно відображали сутність шкідливої активності.

Порівняльний аналіз обох систем засвідчив, що застосування машинного навчання значно підвищує інформативність, структурованість та зручність подальшого аналізу зібраних даних. Гібридна система зберігає сильні сторони сигнатурного аналізу, але усуває найбільш проблемні аспекти, пов’язані з надлишковістю даних. Після обробки ML-моделлю лог-файли стають значно чистішими, що дозволяє сфокусуватися на ключових аномаліях, покращити швидкість реагування та точність інтерпретації подій. Таким чином, комбінований підхід демонструє значний потенціал для використання у реальних системах захисту корпоративних мереж.

Загалом результати проведеного дослідження підтверджують, що інтеграція моделей машинного навчання у системи виявлення вторгнень є перспективним і ефективним напрямом розвитку IDS-технологій. Отримані

результати можуть бути використані для вдосконалення існуючих рішень у сфері кіберзахисту, зменшення навантаження на аналітиків безпеки та підвищення точності виявлення складних атак. У майбутньому подібні системи можуть бути розширені за рахунок використання більш складних моделей, таких як нейронні мережі або моделі глибокого навчання, а також шляхом додаткової автоматизації реагування на інциденти.

Таким чином, поставлені у роботі завдання були успішно виконані, а мета дослідження досягнута. Розроблена система показала при моделюванні її роботи високу ефективність та може бути рекомендована як основа для подальших досліджень і практичного застосування у системах захисту мережевих інфраструктур.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. J. Green. Security Architecture: A Practical Guide to Designing Proactive and Resilient Cyber Protection. BCS, The Chartered Institute for IT, 2025. 358 p.

2. Wireless Communication Security (Advances in Data Engineering and Machine Learning) / edited by Manju Khari et al. Wiley-Scrivener, 2023. 288 p.

3. Talukder, M.A., Islam, M.M., Uddin, M.A. et al. Machine learning-based network intrusion detection for big and imbalanced data using oversampling, stacking feature embedding and feature extraction. Journal of Big Data, 11, 33 (2024). DOI: https://doi.org/10.1186/s40537-024-00886-w

4. Тимощук, В., Ванца, В., Карнаухов, А., Орловська, А., Тимощук, Д. (2024). Порівняльний аналіз підходів до виявлення вторгнень, заснованих на сигнатурах та аномаліях. Матеріали конференції MCND (29 листопада 2024 р.; Житомир, Україна), с. 328–332.

5. Thomas L. Case Enterprise Networks: Infrastructure & Security. Prospect Press, 2025. 558 p.

6. Joseph Migga Kizza. Guide to Computer Network Security. Springer Nature Switzerland AG, 2024. 646 p.

7. Тимощук, Д., Ясний, О., Митник, М., Загородна, Н., Тимощук, В. (2024). Виявлення та класифікація DDoS-атак методами машинного навчання. CEUR Workshop Proceedings, 3842, с. 184–195.

8. M.H. Bhuyan, D.K. Bhattacharyya, J.K. Kalita. Network Traffic Anomaly Detection and Prevention: Concepts, Techniques and Tools. Springer International Publishing AG, 2017. 263 p.

9. H. A. Salman, A. Kalakech, & A. Steiti. Random Forest Algorithm Overview. Babylonian Journal of Machine Learning, 2024, pp. 69–79. DOI: https://doi.org/10.58496/BJML/2024/007

10. Ahmed, U., Nazir, M., Sarwar, A. et al. Signature-based intrusion detection using machine learning and deep learning approaches empowered with fuzzy

clustering. Scientific Reports, 15, 1726 (2025). DOI: https://doi.org/10.1038/s41598-025-85866-7

11. Parag Deoskar, Ajay Kumar Sachan. Enhancing intrusion detection systems using hybrid deep learning models. International Journal of Cloud Computing and Database Management, 6(1):29–42. DOI: https://doi.org/10.33545/27075907.2025.v6.i1a.82

12. S. A. H. Moamin, M. K. Abdulhameed, R. M. Al-Amri, A. D. Radhi, R. K. Naser, & L. G. Pheng. Artificial Intelligence in Malware and Network Intrusion Detection: A Comprehensive Survey of Techniques, Datasets, Challenges, and Future Directions. Babylonian Journal of Artificial Intelligence, 2025, pp. 77–98. DOI: https://doi.org/10.58496/BJAI/2025/008

13. Machine and Deep Learning Solutions for Intrusion Detection and Prevention in IoTs: A Survey/ P. L. S. Jayalaxmi at all. IEEE Access (Volume: 10), 2022. 121173 – 121192 p. DOI: 10.1109/ACCESS.2022.3220622

14. M. Collins Network Security Through Data Analysis: From Data to Action, 2nd Edition. O'Reilly Media, 2017. 428 p.

15. Machine Intelligence and Big Data Analytics for Cybersecurity Applications/ Editors: Yassine Maleh, Mohammad Shojafar, Mamoun Alazab, Youssef Baddi. Springer Nature Switzerland AG, 2021. 539 p.

16. D. Minoli and B. Occhiogrosso AI Applications to Communications and Information Technologies: The Role of Ultra Deep Neural Networks. Wiley-IEEE Press, 2023. 496 p.

17. M. Scarfone, P. Mell. Guide to Intrusion Detection and Prevention Systems (IDPS). NIST Special Publication 800-94, 2007. 121 p.

18. S. Axelsson. Intrusion Detection Systems: A Survey and Taxonomy. Chalmers University of Technology, 2000. 56 p.

19. G. Kim, S. Lee, S. Kim. A Novel Hybrid Intrusion Detection Method Integrating Anomaly Detection with Misuse Detection. Expert Systems with Applications, 41(4), 2014, pp. 1690–1700. DOI: 10.1016/j.eswa.2013.08.066

20. S. M. Mousavi, M. St-Hilaire. Early detection of DDoS attacks using neural networks and naive Bayes classification. Computer Communications, 67, 2015, pp. 30–38.

21. R. Mitchell, I. R. Chen. A Survey of Intrusion Detection Techniques for Cyber-Physical Systems. ACM Computing Surveys, 46(4), 2014, pp. 1–29. DOI: 10.1145/2542049

22. Y. Xin et al. Machine Learning and Deep Learning Methods for Cybersecurity. IEEE Access, 6, 2018, pp. 35365–35381. DOI: 10.1109/ACCESS.2018.2836950

23. F. Haddadi, G. Ghorbani. A Graph-Based Feature Selection Approach for Network Intrusion Detection. IEEE Transactions on Cybernetics, 48(12), 2018, pp. 3235–3247.

24. H. Hindy et al. A Taxonomy of Network Threats and the Deep Learning Methods Used for Intrusion Detection. IEEE Communications Surveys & Tutorials, 23(2), 2021, pp. 1383–1412.

25. D. Dua, C. Graff. UCI Machine Learning Repository – IDS Datasets. University of California, Irvine, 2020.

26. D. Ulyanov, M. Kharchenko, S. Malygina. Intrusion Detection in Software-Defined Networks Using Deep Learning Classification Models. Information and Software Technology, 151, 2023. DOI: 10.1016/j.infsof.2022.106986

ДОДАТОК А

Лістинг А.1 – Зміст файлу з локальними правилами local.rules

```
# LOCAL lab IDS rules - tuned thresholds to reduce repeated alerts
# 1) Fast portscan (many SYNs) — тріггер коли >=50 SYNs за 10s (by_src)
alert tcp any any -> any any (msg:"LOCAL Fast Portscan (many SYNs)"; flags:S; threshold:ty>
# 2) Slow portscan (low-rate) — тріггер коли >=200 різних SYN портів за годину (повільніши>
alert tcp any any -> any any (msg:"LOCAL Slow Portscan (low rate)"; flags:S; threshold:typ>
# 3) SSH brute-force — багато спроб до порту 22 (помірніший поріг)
alert tcp any any -> any 22 (msg:"LOCAL SSH Brute Force - multiple attempts"; flow:to_serv>
# 4) RDP brute-force — багато спроб на 3389
alert tcp any any -> any 3389 (msg:"LOCAL RDP Brute Force - multiple attempts"; flow:to_se>
# 5) HTTP SQLi-like (less noisy PCRE + check) — включає декілька типових патернів, але з у>
alert http any any -> any any (msg:"LOCAL HTTP SQLi-like pattern"; http.uri; pcre:"/(\%27)>
# 6) DoS / volumetric spike - tuned to reduce duplicates
alert tcp any any -> any any (msg:"LOCAL Possible volumetric spike (many flows)"; flow:sta>
# 7) UDP flood to DNS (example) — limit 1 alert на dst за 5 хв, високий count
alert udp any any -> any 53 (msg:"LOCAL UDP volumetric spike to DNS port"; threshold:type >
```