МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Західноукраїнський національний університет
Навчально-науковий інститут інноваційних освітніх технологій
Кафедра інформаційно-обчислювальних систем і управління

Василічишена Валерія Сергіївна

Модуль IDS для локальної мережі з комбінованим сигнатурним і поведінковим аналізом трафіку / IDS Module for a Local Network with Combined Signature and Behavioral Traffic Analysis

Спеціальність 122 – Комп’ютерні науки
Освітньо-професійна програма – Комп’ютерні науки

Кваліфікаційна робота

Виконала студентка групи КНз-41
В. С. Василічишена

Науковий керівник:
к.т.н., доцент, О.Р. Осолінський

Кваліфікаційну роботу допущено до захисту
«___»_____ 2026 р.

Завідувач кафедри
_____ Н.В. Дзюбановська

Тернопіль – 2026

Навчально-науковий інститут новітніх освітніх технологій
Кафедра інформаційно-обчислювальних систем і управління
Освітній ступінь «бакалавр»
спеціальність: 122 – Комп'ютерні науки
освітньо-професійна програма – Комп'ютерні науки

ЗАТВЕРДЖУЮ
В.о. завідувача кафедри
_____ Н.В. Дзюбановська
«____» _____20__ р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТЦІ
ВАСИЛІЧИШЕНІЙ Валерії Сергіївні
(прізвище, ім'я, по батькові)

1. Тема кваліфікаційної роботи:
Модуль IDS для локальної мережі з комбінованим сигнатурним і поведінковим аналізом трафіку / IDS Module for a Local Network with Combined Signature and Behavioral Traffic Analysis
керівник роботи О.Р. Осолінський, к.т.н., доцент
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджений наказом по університету від 01 грудня 2025 року № 894.

2. Строк подання студентом закінченої кваліфікаційної роботи 25 травня 2026 р.

3. Вихідні дані до кваліфікаційної роботи: завдання на кваліфікаційну роботу студента, наукові статті, технічна література.

4. Основні питання, які потрібно розробити:
- проаналізувати застосування IDS/IPS, їх класифікацію, моделі загроз та методи виявлення вторгнення для локальної мережі;
- дослідити відкриті програмні платформи для IDS/IPS;
- сформувати постановку задачі;
- розробити функціональну структуру системи та сценарії використання;
- розробити алгоритмічне забезпечення програмного модуля
- розробити інформаційне забезпечення програмного модуля IDS;
- провести вибір та аналіз програмно-апаратних засобів для розробки модуля;
- реалізувати програмно-технологічне забезпечення модуля;
- провести тестування модуля.

5. Перелік графічного матеріалу в роботі:
- загальна архітектура модуля IDS;
- схема функціонування модуля IDS.

6. Консультанти розділів кваліфікаційної роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		Завдання видав	Завдання прийняв

7. Дата видачі завдання 01 грудня 2025 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів кваліфікаційної роботи	Примітка
1	Огляд літературних джерел відповідно до предметної області та складання плану роботи	до 01.01.2026 р.	
2	Написання 1 розділу кваліфікаційної роботи	до 01.02.2026 р.	
3	Написання 2 розділу кваліфікаційної роботи	до 15.03.2026 р.	
4	Написання 3 розділу кваліфікаційної роботи	до 01.05.2026 р.	
5	Представлення попереднього варіанту кваліфікаційної роботи, перевірка та внесення змін керівником	до 15.05.2026 р.	
6	Опрацювання зауважень та представлення завершеного варіанту кваліфікаційної роботи. Підготовка супроводжуючих документів	до 25.05.2026 р.	
7	Перевірка кваліфікаційної роботи на оригінальність тексту	до 30.05.2026 р.	
8	Оформлення кваліфікаційної роботи та отримання допуску до захисту	до 10.06.2026 р.	
9	Подання кваліфікаційної роботи до захисту на засіданні атестаційної комісії	до 10.06.2026 р.	

Студентка_____ В. С. Василічишена
(підпис) (прізвище та ініціали)

Керівник кваліфікаційної роботи_____ О.Р. Осолінський
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Пояснювальна записка до кваліфікаційної роботи: 63 с., 35 рис., 4 таблиці, 4 додатки, 32 джерела.

Метою роботи є розробка модуля IDS для локальної мережі з комбінованим сигнатурним і поведінковим аналізом мережевого трафіку.

В роботі використано методи сигнатурного аналізу мережевого трафіку, виявлення аномалій, екстракції ознак, класифікації на основі штучної нейронної мережі, кореляції подій, а також методи структурованого збереження і візуалізації даних. Реалізація виконана на базі платформи Raspberry Pi 4B, системи Suricata, мови програмування Python, бібліотек TensorFlow і Keras, бази даних SQLite, мікрофреймворку Flask та середовища Oracle VM VirtualBox з дистрибутивом Debian.

Запропоновано архітектуру та програмну реалізацію модуля IDS, який поєднує сигнатурну перевірку подій Suricata з поведінковим аналізом статистичних характеристик трафіку. Реалізовано механізм зчитування та парсингу журналів `eve.json`, екстракцію ознак, формування вектора параметрів і обчислення ймовірності атаки за допомогою нейронної мережі. Передбачено збереження подій у базі даних, класифікацію станів мережі.

Проведено тестування системи на сценаріях нормального мережевого трафіку та імітації атаки, що дозволило підтвердити коректність роботи сигнатурного і поведінкового модулів, правильність запису подій у базу даних та працездатність веб-сервісу моніторингу.

Ключові слова: IDS, ЛОКАЛЬНА МЕРЕЖА, SURICATA, PYTHON, МАШИННЕ НАВЧАННЯ, НЕЙРОННА МЕРЕЖА, ВИЯВЛЕННЯ АНОМАЛІЙ, МЕРЕЖЕВИЙ ТРАФІК, SQLITE, FLASK.

ANNOTATION

Explanatory note to the qualification thesis: 63 pages, 35 figures, 4 tables, 4 appendices, 32 references.

The purpose of the thesis is to develop an IDS module for a local network with combined signature-based and behavioral analysis of network traffic.

The work uses methods of signature-based network traffic analysis, anomaly detection, feature extraction, classification based on an artificial neural network, event correlation, as well as methods of structured data storage and visualization. The implementation is based on the Raspberry Pi 4B platform, the Suricata system, the Python programming language, the TensorFlow and Keras libraries, the SQLite database, the Flask microframework, and the Oracle VM VirtualBox environment with the Debian distribution.

An architecture and software implementation of an IDS module are proposed, combining Suricata signature-based event inspection with behavioral analysis of statistical traffic characteristics. A mechanism for reading and parsing eve.json logs, feature extraction, parameter vector formation, and attack probability estimation using a neural network has been implemented. The system provides event storage in a database and classification of network states.

The system was tested using scenarios of normal network traffic and simulated attacks, which made it possible to confirm the correct operation of the signature-based and behavioral modules, the correctness of event recording into the database, and the functionality of the web monitoring service.

Keywords: IDS, LOCAL NETWORK, SURICATA, PYTHON, MACHINE LEARNING, NEURAL NETWORK, ANOMALY DETECTION, NETWORK TRAFFIC, SQLITE, FLASK.

ЗМІСТ

ВСТУП

В сучасних локальних мережах постійно зростає кількість підключених пристроїв і сервісів, а разом із цим збільшується обсяг мережевого трафіку. Тому локальна мережа вже не є лише допоміжним елементом, а безпосередньо впливає на стабільність роботи користувачів, серверів і прикладних сервісів. Одночасно збільшується кількість загроз: спроби несанкціонованого доступу, поширення шкідливого програмного забезпечення, викрадення даних і атаки на доступність мережевих служб.

Вразливості програмного забезпечення, помилки конфігурації, слабкі механізми автентифікації, а також дії внутрішніх і зовнішніх порушників створюють постійну небезпеку для локальних мереж. Особливу проблему становить те, що частина атак може тривалий час залишатися непоміченою, маскуватися під звичайну активність користувачів або використовувати легітимні мережеві служби для досягнення шкідливої мети. Тому в такій ситуації традиційних засобів захисту, таких як міжмережеві екрани чи антивірусні програми, часто недостатньо, оскільки вони не завжди дають змогу вчасно виявити складні, розподілені або нові типи вторгнень.

Одним із практичних засобів контролю безпеки локальної мережі є використання систем IDS та IPS. Такі системи дають змогу аналізувати мережевий трафік, виявляти ознаки атак, фіксувати підозрілу активність, формувати сповіщення для адміністратора та, в разі потреби, автоматично блокувати небезпечні дії. Їх використання доцільне в локальних мережах, де необхідно постійно контролювати обмін даними між вузлами, серверами та користувацькими пристроями, а також швидко реагувати на інциденти безпеки. В результаті адміністратор отримує більше інформації про стан мережі та може швидше реагувати на підозрілу активність. Сучасні IDS та IPS можуть працювати на основі сигнатурного аналізу, виявлення аномалій або політик безпеки. Кожен підхід ефективний в різних умовах: сигнатури краще працюють з відомими атаками, а поведінковий аналіз допомагає відмічати нетипову активність. Сигнатурний аналіз

забезпечує точне виявлення вже відомих загроз, поведінкові методи дають змогу реагувати на нетипову активність і потенційно виявляти нові атаки, а політико-орієнтований підхід дозволяє контролювати дотримання встановлених правил доступу та взаємодії в мережі. В практичних умовах це вимагає правильного вибору моделі захисту, програмної платформи та способу інтеграції системи в локальну мережу.

Розвиток відкритих платформ, зокрема Snort, Suricata та Zeek, дозволяє будувати систему моніторингу без придбання дорогих комерційних рішень. Такі інструменти підтримують аналіз пакетів, правила виявлення, журналювання подій, інтеграцію з системами візуалізації та адміністрування, що робить їх придатними як для навчальних цілей, так і для практичного використання в організаціях різного масштабу. На основі них можна створити систему, яка враховує особливості конкретної локальної мережі та потреби адміністратора.

Метою кваліфікаційної роботи є розробка модуля IDS для локальної мережі з комбінованим сигнатурним і поведінковим аналізом трафіку, який забезпечує виявлення підозрілої активності, оцінювання ймовірності атаки, збереження подій у базі даних та відображення результатів у веб-інтерфейсі.

Завдання кваліфікаційної роботи полягають у послідовному виконанні наступних етапів:

– провести аналіз предметної області систем виявлення та запобігання вторгненням у локальних мережах;

– провести огляд і аналіз існуючих IDS/IPS-рішень та відкритих платформ Suricata, Snort і Zeek;

– сформувати постановку задачі розробки IDS-модуля для локальної мережі;

– розробити функціональну структуру системи та визначити сценарії її використання;

– розробити архітектуру модуля IDS з урахуванням сигнатурного та поведінкового аналізу мережевого трафіку;

– розробити алгоритмічне забезпечення модуля, включаючи захоплення трафіку, сигнатурну перевірку, екстракцію ознак, класифікацію подій і кореляцію результатів;

– розробити інформаційне забезпечення, сформувати структуру вхідних і вихідних даних, інфологічну модель бази даних, описати сутності, зв’язки, потоки даних і логіку збереження інформації про інциденти;

– провести вибір та обґрунтування програмно-апаратних засобів для реалізації IDS-модуля;

– реалізувати програмно-технологічне забезпечення модуля IDS;

– провести тестування роботи системи на сценаріях нормального мережевого трафіку та імітації атак, перевірити коректність спрацювань, запис подій у базу даних і відображення результатів у веб-інтерфейсі.

Об’єкт дослідження – інформаційні процеси моніторингу, аналізу та виявлення мережевих загроз у локальній мережі.

Предмет дослідження – методи проєктування, реалізації та тестування модуля IDS для локальної мережі на основі поєднання сигнатурного аналізу та поведінкового виявлення аномалій.

Результати кваліфікаційної роботи апробовані та опубліковані в матеріалах X Міжнародної студентської наукової конференції «Діджиталізація науки як виклик сьогодення» (м. Миколаїв, Україна).

Кваліфікаційна робота складається із вступу, трьох розділів, висновків, списку використаних джерел та додатків.

1 СТАН ТА АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Поняття IDS/IPS, класифікація, модель загроз та методи виявлення вторгнення для локальної мережі

З моменту появи комп'ютерів люди намагалися знайти способи використовувати помилки апаратного та програмного забезпечення і неправильні налаштування для задоволення гордості та отримання прибутку, а також з ідеологічних і політичних причин. В останнє десятиріччя спостерігається ускладнення методів кіберзагроз, що проявляється у зростанні кількості атак на доступність, поширенні програм-вимагачів та збільшенні ризиків компрометації даних, що підсилює потребу у використанні систем IDS/IPS [1]. В таких умовах при швидкому розвитку технологій шифрування трафіку і в загальному збільшені мережі питання про появу технологій які зможуть виявляти а також запобігати спробам вторгнень в мережу стояло першочергово. У сучасних роботах з виявлення аномалій у системах кібербезпеки до типових прикладів підозрілої поведінки відносять нетипові спроби входу, незвичні передавання даних, несанкціонований доступ та інші відхилення від нормального профілю мережевої активності [1]:

1. Спроба вторгнення – хтось, хто намагається вторгнутися в систему, може генерувати ненормально високий рівень невдалих спроб введення пароля щодо одного облікового запису або системи в цілому.

2. Маскування або успішне вторгнення – хтось, хто входить в систему через неавторизований обліковий запис і пароль, може мати інший час входу, місцезнаходження або тип з'єднання, ніж законний користувач облікового запису. Крім того, поведінка зловмисника може значно відрізнятися від поведінки законного користувача. Зловмисник може проводити більшу частину часу, переглядаючи каталоги та виконуючи команди стану системи, тоді як законний користувач може зосередитися на редагуванні або компіляції та зв'язуванні програм.

3. Вторгнення законного користувача – користувач, який намагається проникнути в механізми безпеки операційної системи, може виконувати різні

програми або викликати більше порушень захисту через спроби отримати доступ до несанкціонованих файлів або програм. Якщо його спроба буде успішною, він отримає доступ до команд і файлів, які зазвичай йому недоступні.

4. Витік інформації законним користувачем – користувач, який намагається викрасти конфіденційні документи, може входити в систему в незвичайний час або направляти дані на віддалені принтери, які зазвичай не використовуються.

5. Виведення інформації легітимним користувачем – користувач, який намагається отримати несанкціоновані дані з бази даних шляхом агрегації та висновку, може отримати більше записів, ніж зазвичай.

6. Троянський кінь – поведінка троянського коня, вбудованого в програму або заміненого нею, може відрізнятися від поведінки законної програми за часом роботи процесора або активністю вводу/виводу (1/0).

7. Вірус – вірус, вбудований в систему, може спричинити збільшення частоти перезапису виконуваних файлів, використання пам'яті виконуваними файлами або виконання певної програми в міру поширення вірусу.

8. Відмова в обслуговуванні – зловмисник, який може монополізувати ресурс (наприклад, мережу), може мати аномально високу активність щодо цього ресурсу, тоді як активність всіх інших користувачів буде аномально низькою [2].

На основі цих даних розпочалась розробка таки моделей як IDS/IPS які стали рішенням багатьох проблем а також дали поштовх розвитку подібним систем.

Перші концептуальні засади систем виявлення вторгнень були сформульовані на початку 1980-х років у межах досліджень, присвячених моніторингу загроз комп’ютерній безпеці для військових і спеціалізованих інформаційних систем [3]. У цих роботах було запропоновано підхід до виявлення загроз на основі аналізу подій та статистичних відхилень у поведінці користувачів і системи. Подальший розвиток цієї ідеї привів до практичної реалізації систем мережевого моніторингу безпеки наприкінці 1980-х — на початку 1990-х років [4]. Надалі розвиток IDS зумовив формування кількох основних напрямів побудови систем виявлення вторгнень, серед яких виокремлюють такі системи:

Виявлення вторгнень в мережу (NID) – система працює по принципу передачі інформації між хостами. Зазвичай пристрої виявлення вторгнень в мережу, які називають «аналізатор трафіку», перехоплюють пакети, що передаються різними засобами зв'язку та протоколами, в більшості через TCP/IP. Після перехоплення пакети аналізуються різними способами. Деякі пристрої NID просто порівнюють пакет із базою даних сигнатур, що складається з відомих атак і «відбитків» зловмисних пакетів, а інші шукають аномальну активність пакетів, яка може вказувати на зловмисну поведінку [5].

Виявлення вторгнень на основі хоста (HID) – такі системи уже базуються на основі для моніторингу, виявлення та реагування на активність користувачів і системи, а також на атаки на певний хост. Деякі більш надійні інструменти також пропонують управління політикою аудиту та централізацію, надають дані для криміналістичної експертизи та статистичний аналіз а в деяких випадках забезпечують певний контроль доступу [5].

Різницею між виявленням вторгнень на основі хоста та на основі мережі полягає в тому, що NID має справу з даними, що передаються від хосту до хосту, тоді як HID займається тим, що відбувається на самих хостах. З часом такий тип розділу ставав не ефективним через швидкість розвитку мереж і вдосконалення способів вторгнення. Тому на основі NID та HID створили їх гібрид з централізованим управлінням виявленням вторгнень. Таким рішенням стало створення мережевих вузлів (NNID). Головною їх задачею стало усунення недоліків, притаманних традиційним системам NID. Мережевий вузол витягує технологію перехоплення пакетів з кабелю і розміщує її на хості. З NNID розташовується таким чином, що він перехоплює пакети після того, як вони досягають кінцевої мети. Потім пакет аналізується так, ніби він проходить по мережі через звичайний аналізатор трафіку [5].

Подальшим розвитком традиційні можливості IDS є система запобігання вторгненням (IPS). Ця система успадковує базовий функціонал традиційних IDS щодо виявлення загроз та реєстрації інцидентів, проте додатково забезпечує механізми автоматизованого запобігання атакам. Перехід на нові моделі масово

почався у 2003 році. Популярність IPS зумовлена також активним розвитком бездротової мережі, що забезпечувало миттєве реагування на загрозу і блокувало її до того як вона потрапляє до користувача на пристрій [6].

Загалом функціонування будь-якої системи IDS/IPS визначається застосованим методом аналізу мережевих пакетів. Сучасні системи за типом аналітичного ядра поділяються на дві основні категорії: сигнатурні та поведінкові (аномальні), до яких також часто додають методи на основі політик безпеки. Кожен із цих підходів характеризується власними принципами роботи, сферами використання та певними обмеженнями [7].

1.1.1 Сигнатурний метод виявлення вторгнення

Виявлення на основі сигнатур аналізує мережеві пакети на наявність атак унікального характеру або поведінки, пов'язаних із конкретною загрозою. Прикладом сигнатури атаки є послідовність коду, що з'являється в певному варіанті шкідливого програмного забезпечення. На основі цих сигнатур ведеться база даних з якими порівнюються мережеві пакети та йде реагування на атаку. Бази даних сигнатур необхідно регулярно оновлювати новою інформацією про загрози, тому що з'являються нові кібератаки, а існуючі атаки еволюціонують. Але абсолютно нові атаки, які ще не проаналізовано на наявність сигнатур, можуть уникнути перевірки. Тому цей метод має низький ризик хибного реагування на загрозу, завдяки цьому система спрацьовує тільки на реальні атаки. Також завдяки оптимізаціям, які не затримують трафік можна збільшити швидкість аналізу даних. Ще одною із переваг є деталізація, яка чітко може вказати куди була направлена атака і дозволяє швидко прийняти рішення по протидії.

Незважаючи на всі переваги сигнатурної перевірки є і недоліки та обмеження. Одна з таких вразливостей - це атаки нових і невідомих типів які не занесені в базу даних. Навіть якщо тип атака є в базі, мінімальні зміни в коді вже можуть обійти її, адже збігу не буде виявлено, тому вона потребує постійного оновлення [7].

1.1.2 Виявлення на основі поведінки

Використовують штучний інтелект і машинне навчання для створення та постійного вдосконалення базової моделі нормальної мережевої активності. Порівнюючи поточну мережеву активність з моделлю і реагує, коли виявляє відхилення, наприклад процес, який використовує більше пропускної здатності, ніж зазвичай, або пристрій, який відкриває порт, який зазвичай закритий. Сама модель базується на основі аномалій реагує на будь-яку ненормальну поведінку миттєво, що часто може блокувати нові кібератаки, які можуть уникнути виявлення на основі сигнатур. Також метод може виявляти експлойти нульового дня атаки, які використовують вразливості програмного забезпечення до того, як розробник програмного забезпечення дізнається про них або встигне їх виправити.

Однак метод на основі аномалій можуть бути більш схильні до помилкових спрацьовувань. Навіть нешкідлива діяльність, така як авторизований користувач, який вперше отримує доступ до конфіденційного мережевого ресурсу, може спрацювати як аномалія. В результаті чого авторизовані користувачі можуть бути вигнані з мережі або їх IP-адреси можуть бути заблоковані. Крім того систему можна обманути під час навчання, що може призвести до не спрацьовування на загрозу [7].

1.1.3 Виявлення на основі політик

Цей метод базуються на політиці безпеки, встановлених командою безпеки під час розробки або експлуатації. Кожного разу коли виявляється дія, яка порушує політику безпеки, цю спробу блокують. Наприклад, SOC може встановити політики контролю доступу, які визначають, які користувачі та пристрої можуть отримати доступ до хосту. Якщо неавторизований користувач або користувач який немає доступу до хосту намагається підключитися, метод на основі політик зупиняє його. Завдяки ній забезпечується високий рівень прозорості і передбачень в мережі а також сегментацію по рівнях доступу.

Хоч метод працює непогано в нього є і свої мінуси. А саме складність виявлення атак у дозволених мережах і підтримка у динамічних системах [8].

1.2 Огляд відкритих програмних платформ для IDS/IPS

На сьогодні платформи IDS/IPS поділяються на дві основні категорії: комерційні (Cisco, Palo Alto Networks, Fortinet) та платформи з відкритим кодом. У цей час як комерційні системи використовуються для великих центрів і мають закриту архітектуру, відкриті платформи такі як Snort, Suricata та Zeek дають можливість проектувати свою мережу в рамках навчання або підприємницької діяльності [9].

1.2.1 Suricata

Suricata працює шляхом аналізу мережевого трафіку, застосування заздалегідь визначених правил для виявлення зловмисної діяльності та оповіщення адміністраторів про потенційні загрози. Розміщення в такій ситуації має вирішальне значення для забезпечення повного покриття мережі та ефективного виявлення загроз, а розміщення безпосередньо після брандмауера та перед DMZ є оптимальним для моніторингу зовнішнього трафіку. Така конфігурація ефективно контролює трафік, що надходить з Інтернету, включаючи вхідний та вихідний трафік, що є критично важливим для захисту зовнішніх служб, таких як веб-сервери та поштові сервери. Альтернативним варіантом, хоча і з підвищеною вразливістю до прямого інтернет-трафіку, може бути розміщення Suricata на периферії мережі, перед брандмауером. Це вимагатиме надійних заходів безпеки для захисту від потенційних прямих атак, але може забезпечити раннє виявлення загроз. Внутрішнє розміщення Suricata між мережевим шлюзом, який виконує перетворення мережевих адрес (NAT), та основним комутатором ефективно контролює внутрішній мережевий трафік. Таке розміщення є ключовим для виявлення загроз у мережі, включаючи трафік між VLAN та серверами [10, 11]. На рисунку 1.1 зображено дашборд моніторингу подій безпеки Suricata.

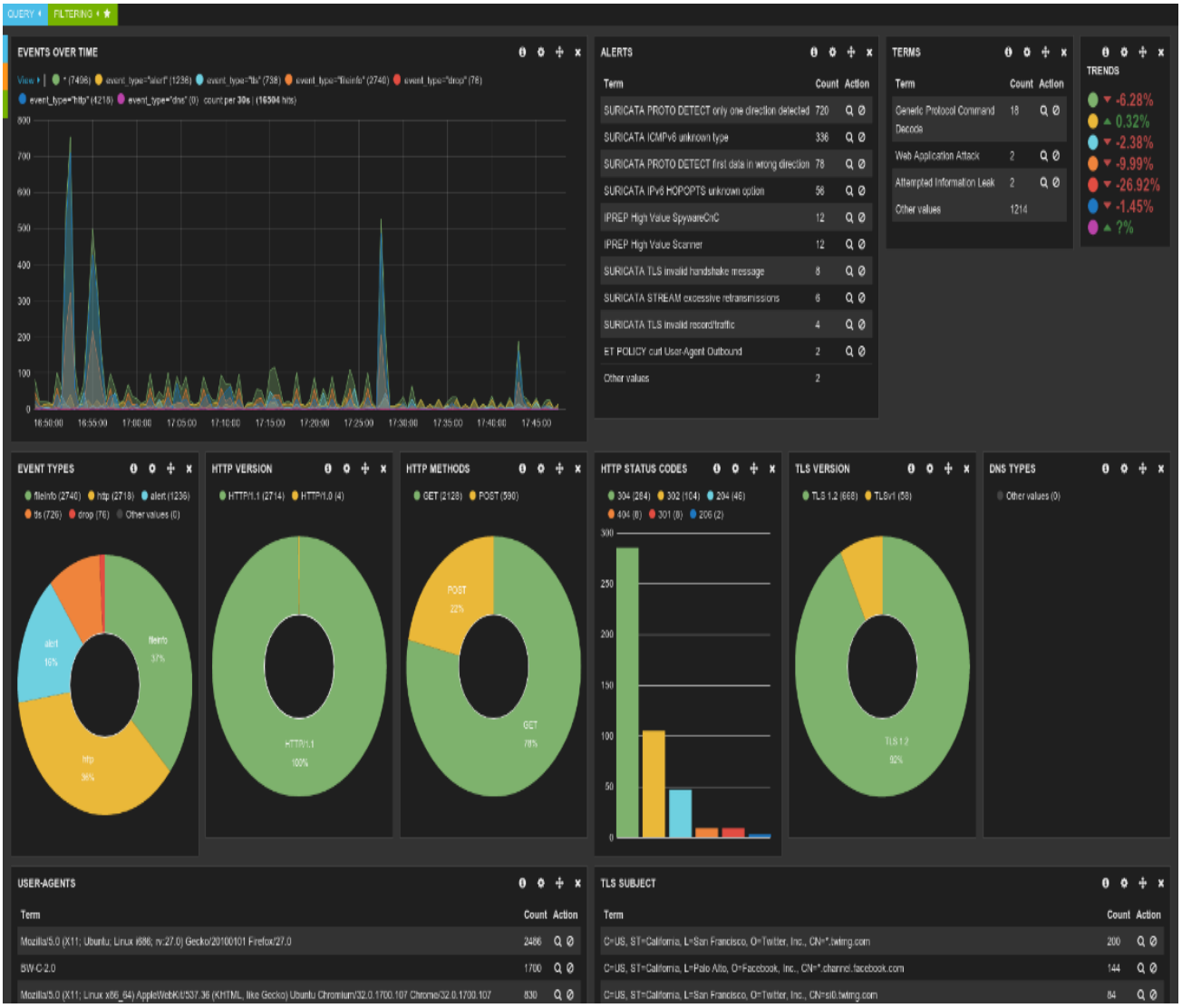


Рисунок 1.1 - Моніторингу подій Suricata

1.2.2 Snort

Принцип роботи Snort є моніторинг мережевого трафіку та застосування заздалегідь визначених правил для виявлення зловмисної діяльності, генеруючи сповіщення для адміністраторів, щоб вони могли вжити відповідних заходів. Його сильна сторона полягає в широкому наборі правил, які можна налаштувати відповідно до конкретних потреб організації в галузі безпеки. Ця адаптивність дозволяє Snort досягти високих результатів в різних середовищах, забезпечуючи індивідуальний захист від широкого спектру загроз.

Snort 3.0 була вдосконалена для більшої складності та точності правил виявлення, що забезпечує додаткові можливості виявлення загроз [10, 12]. На рисунку 1.2 зображено інтерфейс перегляду сповіщень.

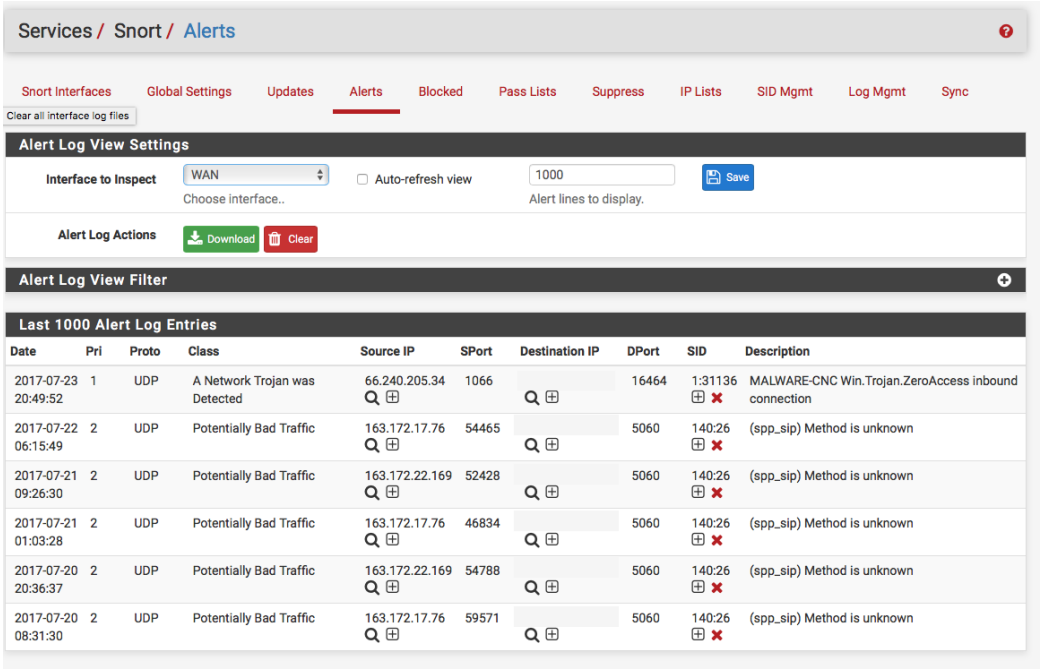


Рисунок 1.2 - Інтерфейс Snort

1.2.3 Zeek

Цей пакет являє собою пасивний аналізатор мережевого трафіку з відкритим кодом. Багато операторів використовують його монітор мережевої безпеки (NSM) для підтримки розслідувань підозрілої або зловмисної діяльності. Zeek також підтримує широкий спектр завдань аналізу трафіку, що виходять за межі сфери безпеки, включаючи вимірювання продуктивності та усунення несправностей.

Перевагою, яку користувач отримує від Zeek, є великий набір журналів, що описують мережеву активність. Ці журнали містять не тільки вичерпний запис кожного з'єднання, яке спостерігається в мережі, але й транскрипти на рівні додатків. До них належать усі сеанси HTTP з запитуваними URI, ключовими заголовками, типами MIME та відповідями сервера, запити DNS із відповідями, сертифікати SSL, ключовий вміст сеансів SMTP та багато іншого. За замовчуванням він записує всю цю інформацію в структуровані файли журналів, розділені табуляцією або JSON, придатні для подальшої обробки за допомогою зовнішнього програмного забезпечення [13]. На рисунку 1.3 панель виявлення загроз Zeek.

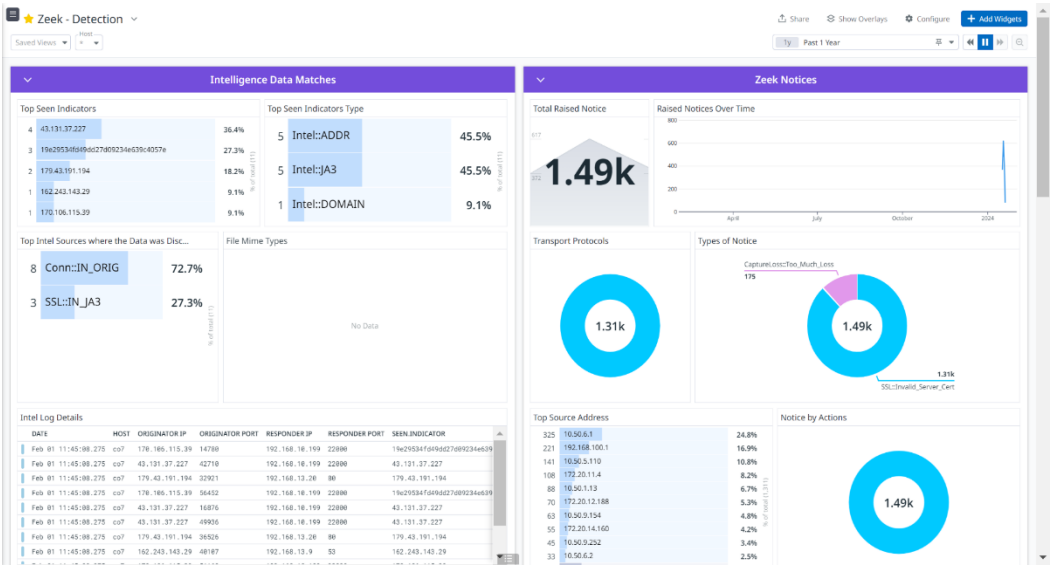


Рисунок 1.3 - Виявлення загроз в Zeek

1.3 Постановка задачі

В сучасних локальних мережах домашнього та корпоративного рівня кількість підключених пристроїв постійно зростає. Крім ПК і смартфонів в мережі часто працюють IP-камери, принтери, телевізори, IoT-пристрої та різні хмарні сервіси, які безперервно генерують трафік. Через це навіть прості атаки можуть бути непомітними на фоні звичайної активності, а підозрілі події проявляються як “дрібні відхилення”, тобто нетипова частота пакетів, сканування портів, підозрілі DNS/TLS-запити, сплески ICMP або незвичні з’єднання між вузлами.

Незважаючи на наявність готових комерційних IDS/IPS-рішень, більшість з них орієнтовані на корпоративний сегмент, потребують окремої інфраструктури, складні в налаштуванні або вимагають достатньо потужного обладнання. В локальних мережах малого масштабу часто немає можливості розгортати “важкі” системи моніторингу, але потреба в контролі та розумінні стану мережі залишається. Водночас чисто сигнатурний підхід добре виявляє відомі атаки, але може пропускати нові або модифіковані загрози, тоді як поведінковий аналіз здатний помічати аномалії, але потребує правильного підбору ознак та порогів, щоб мінімізувати хибні спрацювання.

Актуальність проблеми обумовлена необхідністю створення доступного та практичного IDS-модуля, який може працювати на ресурсно-обмеженій платформі (наприклад Raspberry Pi), аналізувати події у режимі, наближеному до реального часу, накопичувати історію інцидентів та надавати зрозумілий інтерфейс для адміністратора. Оптимальним підходом в такому випадку є поєднання сигнатурного аналізу із поведінковою оцінкою трафіку, що дозволяє отримати більш стійкий результат при зміні профілю мережевої активності.

Метою цієї роботи є розробка архітектури та програмної реалізації модуля IDS для локальної мережі з комбінованим сигнатурним і поведінковим аналізом трафіку, який забезпечуватиме фіксацію подій, визначення стану “АТАКА/NORMA” та відображення результатів у веб-інтерфейсі.

Для досягнення цієї мети необхідно вирішити наступні завдання:

1. Розробити концепцію та архітектуру IDS-модуля для локальної мережі з урахуванням обмежених ресурсів edge-платформи.

2. Виконати аналіз існуючих IDS/IPS-рішень та відкритих платформ (Suricata, Snort, Zeek) і визначити їх роль у побудові системи.

3. Реалізувати сигнатурний аналіз мережевих подій на базі Suricata та механізм збору інцидентів у структурованому вигляді.

4. Реалізувати поведінковий аналіз на основі статистичних ознак трафіку та ML-моделі для оцінки ймовірності атаки.

5. Організувати збереження подій у базі даних та створити веб-інтерфейс для перегляду поточного стану і історії інцидентів.

6. Провести вибір програмно-апаратних засобів для розгортання IDS-сенсора в локальній мережі.

7. Виконати тестування роботи модуля на сценаріях нормального трафіку та імітації атак, перевіривши коректність спрацювань і запис подій.

2 АЛГОРИТМІЧНЕ ТА ІНФОРМАЦІЙНЕ ЗАБЕЗПЕЧЕННЯ МОДУЛЯ IDS ДЛЯ ЛОКАЛЬНОЇ МЕРЕЖІ

2.1 Функціональна структура системи та сценарії використання

Проектування IDS-модуля для SOHO-мережі потребує чіткого визначення його функцій, потоків даних і способів реагування на події. З врахуванням обмежених ресурсів Raspberry Pi система має обробляти трафік з мінімальними затримками та без втрати важливих подій безпеки. Функціональні вимоги для системи класифікують за підсистемами обробки даних:

1. Підсистема моніторингу мережевого трафіку:

- Робота в режимі «Promiscuous Mode» забезпечує повне перехоплення пакетів у мережі, включаючи ті, що не адресувалися самому пристрою;

- Підтримка технологій нульового копіювання (Zero-Copy) для мінімізації на навантажень на процесор використовуючи метод FIFO мережевої карти щоб обробляти трафік до 1 Гбіт/с без втрати пакетів;

- Обробка інкапсульованого трафіку, щоб система могла чітко розпізнавати та обробляти пакети VLAN (стандарт IEEE 802.1Q) (рисунок 2.1).

Рисунок 2.1 - Підсистема моніторинг мережевого трафіку

2. Підсистема декодування та розбору протоколів:

- Розбір протоколів транспортного та мережевого рівнів для декодування з IPv4/IPv6, TCP, UDP, SCTP з метою виявлення аномалій у структурі пакету;

- Збирання потоків для відстеження та об’єднання окремих пакетів в загальний для запобігань атак типу «Evation»;

- Парсинг протоколів прикладного рівня для детальної перевірки протоколів HTTP, DNS, TLS/SSL, SMB, FTP (рисунок 2.2).

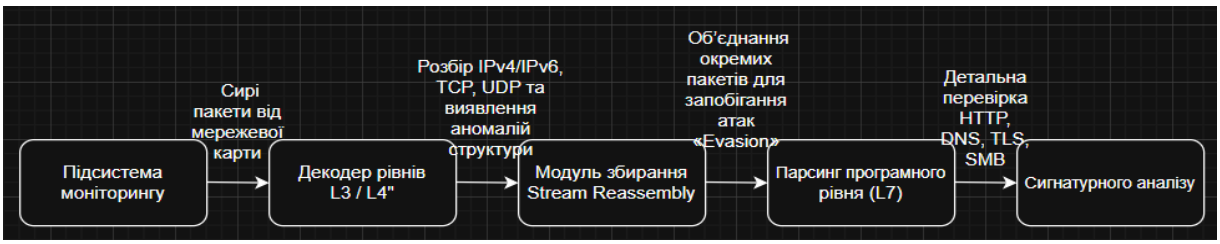


Рисунок 2.2 - Підсистема декодування та розбору протоколів

Підсистема сигнатурного аналізу відповідає за виявлення відомих загроз за попередньо заданими правилами (рисунок 2.3).

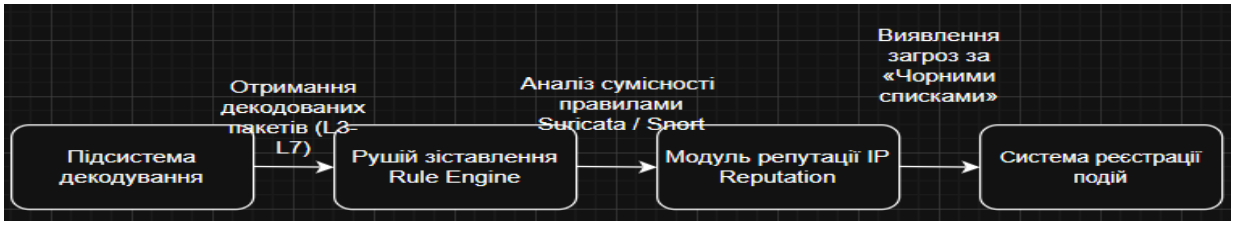


Рисунок 2.3 - Підсистема декодування та розбору протоколів

Вона підтримує стандартні формати правил і є сумісною із синтаксисом правил Suricata або Snort, що дає змогу використовувати готові набори сигнатур та спрощує налаштування системи. Завдяки цьому трафік можна порівнювати з описами відомих атак, шкідливих шаблонів і типових ознак підозрілої активності. Додатково підсистема виконує виявлення загроз на основі репутації IP-адрес і чорних списків IP Blacklists, тобто перевіряє, чи не належить адреса джерела або призначення до переліку відомих небезпечних вузлів. Якщо адреса джерела або призначення збігається з чорним списком, система реєструє подію та показує її адміністратору як потенційну загрозу.

Кореляція подій потрібна для того, щоб не розглядати кожне спрацювання окремо. Система групує пов'язані події та формує один інцидент, який краще відображає загальну картину можливої атаки (рисунок 2.4).

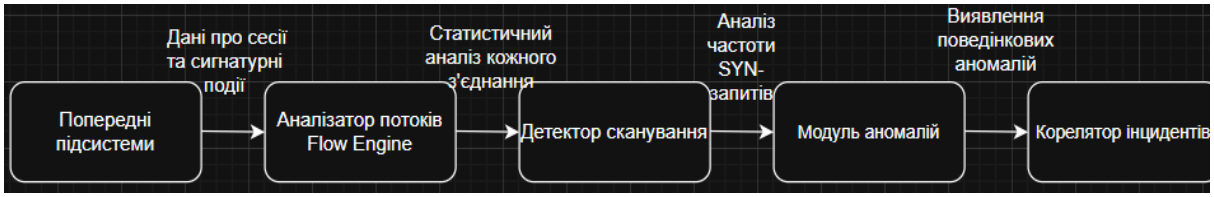


Рисунок 2.4 - Кореляція подій для різних модулів в єдиний інцидент

Для цього виконується статистичний аналіз потоків для кожного з'єднання, під час якого враховуються кількість пакетів, тривалість сеансу, обсяг переданих даних та напрямок обміну, що дозволяє виявляти підозрілі або нетипові мережеві потоки. Окремо здійснюється виявлення мережевого сканування шляхом аналізу частоти SYN-запитів, оскільки надмірна кількість таких запитів за короткий проміжок часу може свідчити про спробу пошуку відкритих портів і доступних мережевих служб. Додатково система виконує виявлення аномалій, порівнюючи поточну активність із типовою поведінкою мережі, що дає змогу визначати незвичні підключення, різке зростання трафіку або інші відхилення, які можуть бути ознаками загрози.

Після виявлення події система зберігає її, присвоює рівень критичності та передає інформацію у веб-інтерфейс для перегляду адміністратором (рисунок 2.5).

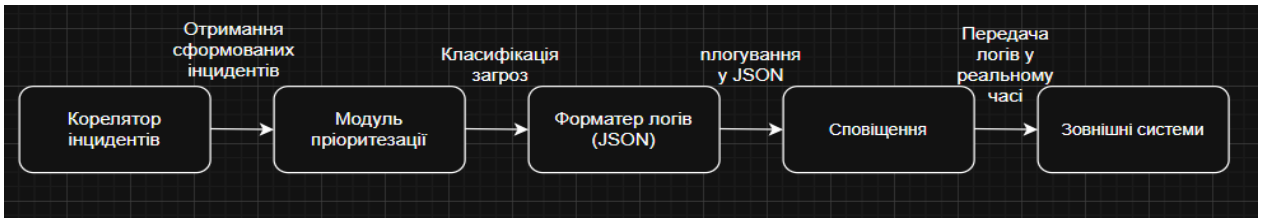


Рисунок 2.5 - Формування звіту та оповіщення у реальному часі

Для цього система виконує структуроване логування у форматі JSON, що дає змогу зберігати події у зручному та стандартизованому вигляді, придатному для автоматичної обробки іншими програмними засобами. Кожній події також може надаватися певний пріоритет, завдяки чому загрози класифікуються за рівнем критичності, а адміністратор отримує можливість швидше відокремлювати найбільш небезпечні інциденти від менш важливих повідомлень. Додатково

система підтримує передачу логів в реальному часі до зовнішніх систем моніторингу, зберігання або візуалізації, що дозволяє централізовано контролювати стан мережі, оперативно виявляти інциденти та своєчасно реагувати на загрози.

2.1.1 Сценарії використання

Сценарій 1 — виявлення відомої атаки. В цьому випадку система спрацьовує на основі правила, яке вже є в базі сигнатур. Вона представлена таблицею 2.1 де описується сценарій виявлення за сигнатурою.

Таблиця 2.1 - Виявлення за сигнатурою

Елементи	Опис
Актори	Зловмисник (ініціатор), Система (виконавець)
Передумови	1. Система працює в штатному режимі 2. База правил (Suricata rules) актуальна 3. Мережевий інтерфейс знаходиться в режимі promiscuous
Основний потік	1.Зловмисник відправляє шкідливий пакет (наприклад, з payload /etc/passwd) у мережу 2. Система перехоплює пакет і декодує протоколи 3. Модуль сигнатурного аналізу порівнює вміст пакету з правилами 4. Система знаходить збіг із правилом (наприклад, ET WEB_SERVER) 5. Система генерує подію (alert) у форматі JSON 6. Подія записується у файл логів eve.json
Альтернативний потік	1. Пакет фрагментований. Система виконує дефрагментацію перед аналізом 2. Трафік зашифрований (HTTPS), і сигнатура не спрацьовує
Результат	Інцидент зафіксовано, адміністратор отримує сповіщення

На основі першого сценарію було створено UML діаграму виявлення за сигнатурою (рисунок 2.6).

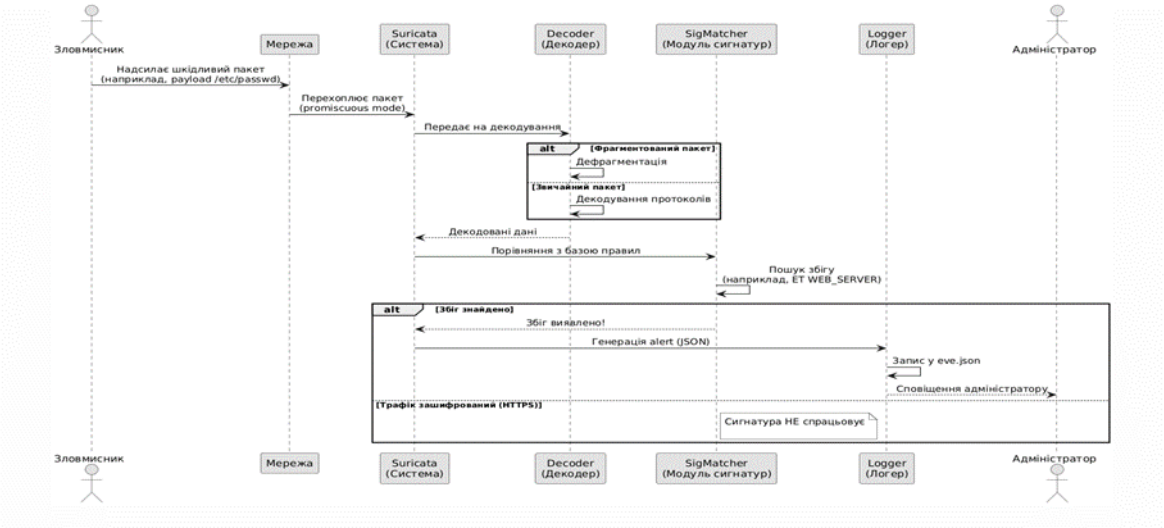


Рисунок 2.6 - UML діаграма виявлення за сигнатурою

Сценарій 2 - Виявлення на основі поведінки. Цей сценарій застосовується тоді, коли сигнатурний аналіз не виявляє загрози, але поведінка трафіку відрізняється від звичайної. Цей спосіб описано в таблиці 2.2 .

Таблиця 2.2 - Виявлення на основі поведінки (аномальної)

Елементи	Опис
Актори	Зловмисник / Заражений хост (ініціатор), Система
Передумови	1. Python-модуль аналітики запущений 2. Сформовано базовий профіль мережі (Baseline) або задано порогові значення
Основний потік	1. Зловмисник починає сканування портів (nmap -sS) або Flood-атаку 2. Система фіксує різке зростання кількості нових TCP-з'єднань за одиницю часу 3. Сигнатурний модуль не бачить шкідливого контенту (пакети порожні, але легітимні) 4. Поведінковий модуль обчислює метрику (, SYN/ACK ratio) 5. Значення метрики перевищує поріг (Threshold) 6. Генерується подія (alert) ANOMALY_DETECTED
Результат	Ідентифіковано підозрілу IP-адресу, навіть без наявності сигнатури атаки

На рисунку 2.7 зображено UML діаграму де зображено сценарій виявлення на основі поведінки.

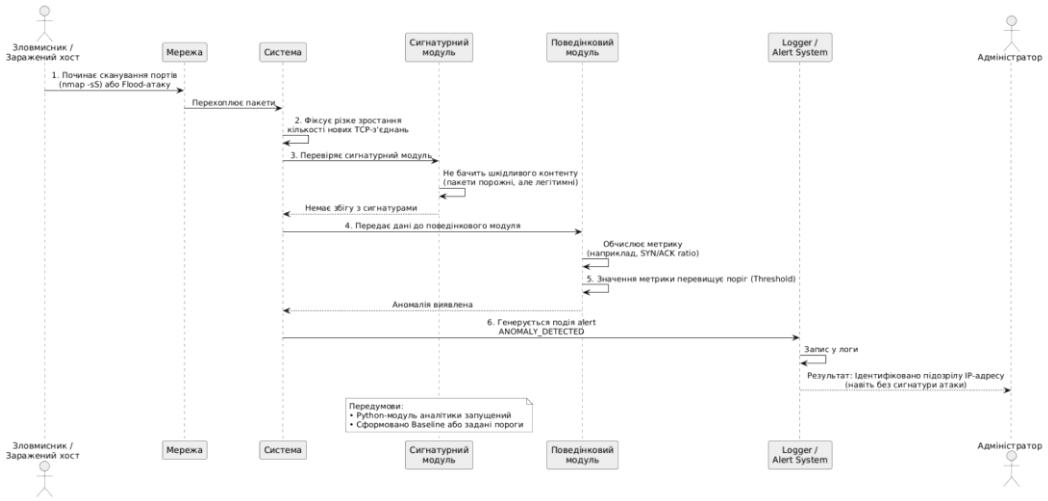


Рисунок 2.7 - UML діаграма виявлення на основі поведінки (аномальної)

Сценарій 3 — перегляд статистики та журналу подій адміністратором. Аналіз представлено в таблиці 2.3 де описується сценарій перегляд журналу подій.

Таблиця 2.3 - Перегляд журналу подій

Елементи	Опис
Актори	Адміністратор
Передумови	1. У базі даних наявні записи про інциденти 2. Адміністратор пройшов автентифікацію
Основний потік	1. Адміністратор відкриває веб-інтерфейс (Dashboard) 2. Система відображає зведену статистику за останні 24 години 3. Адміністратор застосовує фільтр (наприклад, severity: High або src_ip: 192.168.1.50) 4. Система робить вибірку з бази даних і відображає деталі інцидентів (час, тип атаки, payload) 5. Адміністратор експортує звіт у PDF/CSV
Результат	Адміністратор отримав дані для прийняття рішення про блокування хоста або подальшої роботи

На рисунку 2.8 зображено UML діаграму Перегляд журналу подій.

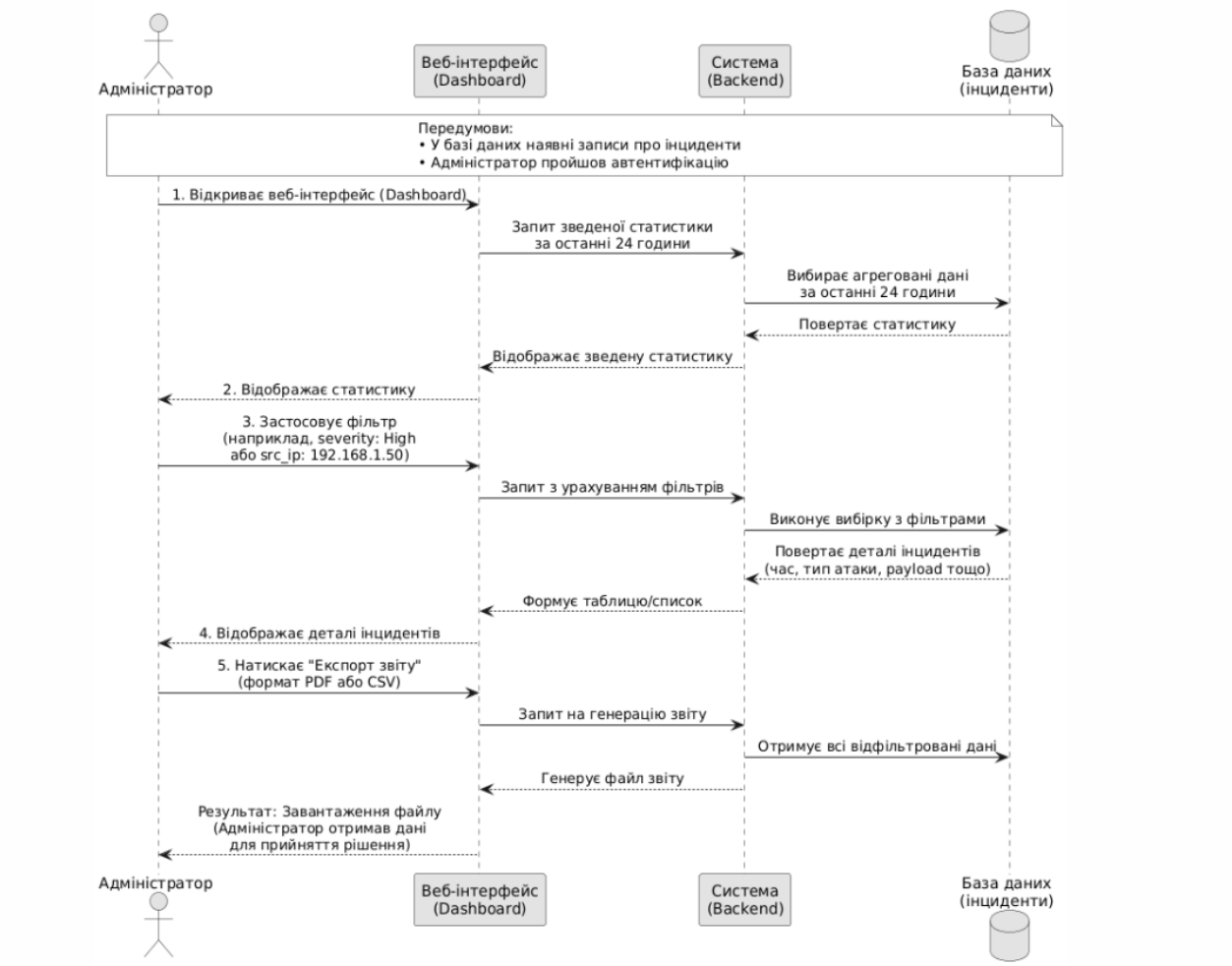


Рисунок 2.8 - UML діаграма перегляду журналу подій

2.2 Алгоритмічне забезпечення програмного модуля

Робота модуля починається з захоплення мережевого трафіку в режимі, наближеному до реального часу. Мережевий інтерфейс фіксує пакети, що проходять через локальну мережу, після чого вони передаються на попередню обробку. Основною задачею цього етапу є підготовка даних до подальшої оброки та декодування. В подальшому система розбирає заголовки протоколів (IP, TCP, UDP), групує пакети для цілісного потоку даних і нормалізує їх. Такий підхід дає змогу аналізувати не окремі пакети, а повну мережеву сесію, що підвищує точність виявлення підозрілої активності. На рисунку 2.9 зображено схему алгоритму захоплення мережевого трафіку.

Рисунок 2.9 - Схема алгоритму захоплення мережевого трафіку

Після отримання інформації система спрямовує їх на два основні етапи перевірки. Перший з них сигнатурний аналіз який порівнює трафік з базою відомих загроз щоб виявити спробу вторгнення. Другим етапом виконується поведінковий аналіз. Він оцінює нетипову активність в мережі та може виявляти загрози, для яких ще немає готових сигнатур. Використовуючи заздалегідь встановлені правила система може працювати, як автономному режимі так і мати можливістю оперативного втручання адміністратора безпеки.

Основне завдання модуля — швидко зафіксувати підозрілу подію та передати інформацію адміністратору. Тому і для цього використовується перевірка в два

етапи. На рисунку 2.10 зображено схему алгоритму роботи сигнатурного аналізу трафіку.

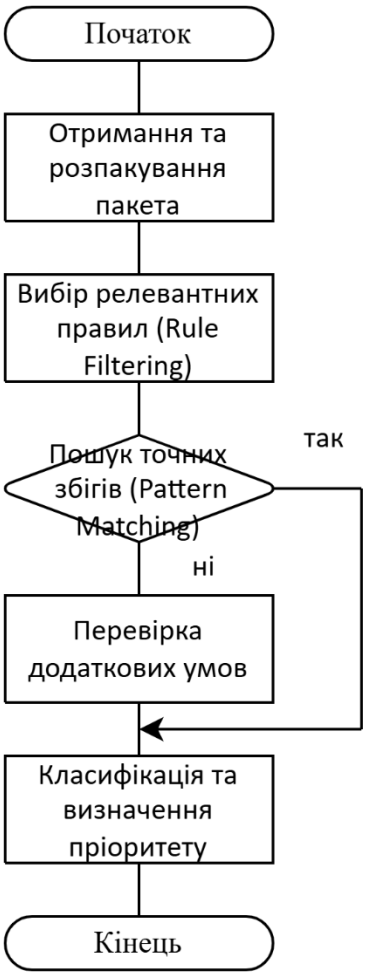


Рисунок 2.10 – Схема алгоритму роботи сигнатурного аналізу

Після отримання та розбору пакетів підсистема порівнює їх із базою відомих сигнатур, тобто характерних ознак атак. Для оптимізації швидкодії система застосовує фільтрацію правил за портами та протоколами. Алгоритм працює за принципом пошуку точних збігів (Pattern Matching), який ідентифікує специфічні послідовності, характерні для SQL-ін’єкцій, переповнення буфера або мережевих троянів. Кожне спрацювання фіксується за унікальним номером правила (SID) та класифікується за рівнем пріоритету: від Priority 1 (критична загроза) до Priority 2 (підозріла активність).

Сигнатурний аналіз не завжди виявляє нові або змінені атаки. Щоб зменшити цей ризик, його можна доповнити поведінковим аналізом, який оцінює відхилення в мережевій активності. На рисунку 2.11 зображено схему алгоритму роботи поведінкового аналізу.

В цьому режимі система шукає відхилення від звичного профілю мережевого трафіку. Коли виявляється суттєве відхилення (аномалія), система одразу генерує попередження про можливу атаку нульового дня. Це важливо для виявлення нових загроз, які ще не внесені до бази сигнатур.

На фінальному етапі відбувається порівняння двох наборів даних від сигнатурної перевірки та поведінкового модуля. Цей метод злиття даних працює за принципом "зваженого голосування". Кожному модулю надається власний коефіцієнт довіри W, а загальний ризик R_total розраховується окремо для хоста або мережевої сесії

$$R_{total} = \frac{\Sigma(Score_i \cdot W_i)}{\Sigma W_i} + Boost_{context}, \quad (2.1)$$

де *Score_i* - оцінка загрози *i*-го модуля, нормалізована в діапазоні від 0 до 1;

W_i - вага або коефіцієнт довіри *i* -го модуля;

Boost_context - додатковий бал, який додається, якщо активи є критичними.

Крім того потрібно адаптувати поріг спрацьовування системи до нормального рівня активності, дозволяючи системі коректно працювати в залежності від часу доби. Вона базуються на методі змінного середнього (Moving Average) та стандартного відхилення.

$$T_{dyn} = \mu_{baseline} + k \cdot \sigma_{baseline}, \quad (2.2)$$

де $\mu_{baseline}$ - середнє значення трафіку (наприклад, пакетів/сек) за останні N днів;

$\sigma_{baseline}$ - стандартне відхилення;

k - коефіцієнт чутливості (2 для середньої чутливості, 3 для консервативної, щоб зменшити FPR).

Після завершення аналізу адміністратор отримує сповіщення у веб-інтерфейсі та приймає рішення: заблокувати IP-адресу, продовжити спостереження або виконати додаткову перевірку. Усі події зберігаються в базі даних, тому їх можна використовувати для перегляду історії, формування звітів і подальшого вдосконалення правил аналізу.

2.3 Інформаційне забезпечення програмного модуля IDS

Програмний модуль IDS для локальної мережі працює з конкретними видами вхідної та вихідної інформації. Ці дані використовуються від перехоплення пакетів до надсилання повідомлення про виявлення загрози і їх запис в базу даних.

Вхідна інформація надходить від мережевого інтерфейсу, який фіксує всі пакети даних в режимі реального часу. Основними є IP-адреси, порти, тип протоколу (TCP/UDP) та корисне навантаження пакета (Payload). Так як трафік може містити як звичайні запити, так і шкідливі фрагменти, для аналізу зберігаються основні характеристики пакетів і потоків.

Вихідною інформацією є статус виявлення вторгнення, ймовірність аномалії, номер сигнатури а також пріоритет загрози та час. В таблиці 2.4 наведено формат вхідної та вихідної інформації.

Таблиця 2.4 - Формат вхідної та вихідної інформації

№	Назва поля	Тип даних	Вимірювання	Призначення
1	src_ip / dst_ip	текст	IPv4/IPv6	Адреси відправника та отримувача
2	src_port / dst_port	ціле число	порти (0–65535)	Порти з'єднання
3	protocol	текст	TCP, UDP, ICMP	Тип мережевого протоколу

Продовження Таблиця 2.4

№	Назва поля	Тип даних	Вимірювання	Призначення
4	tcp_flags	ціле число	бітові прапорці	Прапорці TCP (SYN, ACK) для виявлення сканування
5	payload	масив байтів	текст / hex	Зміст пакета для пошуку сигнатур
6	flow_duration	дійсне число	секунди	Тривалість мережевої сесі
7	flow_iat	дійсне число	мілісекунди	Для детектування DDoS
8	ja3_fingerprint	текст	MD5 хеш	Відбиток клієнта для аналізу шифрованого трафіку
9	anomaly_score	дійсне число	від 0 до 1	Рівень аномальності поведінки
10	signature_sid	ціле число	ID правила	Номер сигнатури, що спрацювала
11	priority	ціле число	1 - 3	Рівень тривоги (1 - критично)
12	timestamp	дата і час	ISO 8601	Час фіксації події

Також для системи необхідно забезпечити логічну структуру зберігання даних. Це дозволить бачити підозрілі події, зберігати статистичні ознаки мережевих потоків для подальшого ML та вести базу заблокованих хостів. Базу даних побудовано за реляційним принципом. Вона містить три основні сутності: Events, Flow_Metrics та Signatures.. Схема зв'язків сутностей представлена на рисунку 2.11.

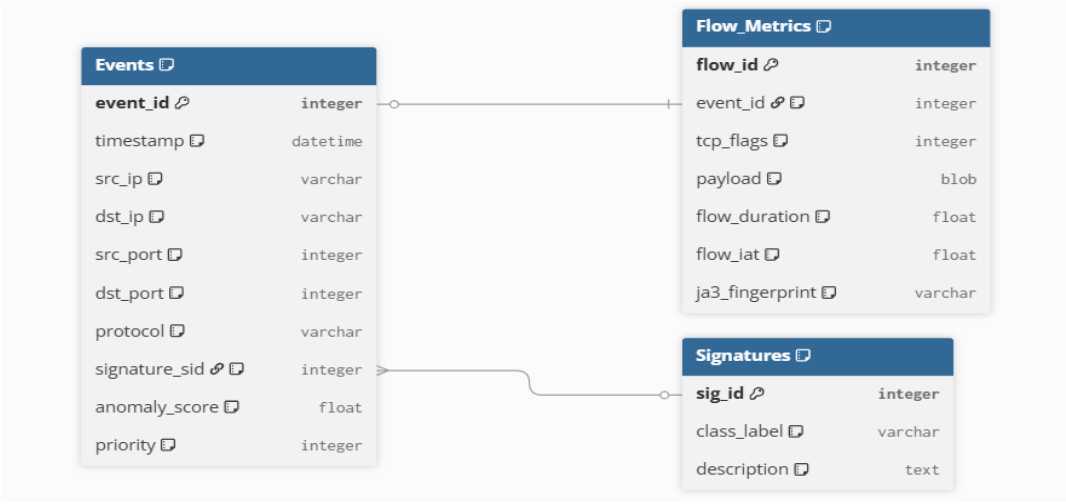


Рисунок 2.11 - ER-діаграма логічної структури бази даних

Основною є таблиця Events, в якій зберігається інформація про кожне зафіксоване спрацювання системи. Також записи мають унікальний ідентифікатор event_id, який слугує первинним ключем. Поле signature_sid є зовнішнім ключем, що посилається на таблицю сигнатур тоді як anomaly_score заповнюється модулем поведінкового аналізу. Додатково використовується таблиця Signatures, яка виконує роль довідника відомих атак. Вона містить текстовий опис загрози та клас атаки та унікальний ідентифікатор правила (sig_id). Це дозволяє уникнути дублювання текстової інформації в основній таблиці подій оптимізуючи базу даних. Така структура спрощує подальше розширення системи, зберігає історію інцидентів і дає змогу аналізувати події за різними параметрами.

3 ПРОГРАМНО-ТЕХНОЛОГІЧНЕ ЗАБЕЗПЕЧЕННЯ СИСТЕМИ

3.1 Вибір та аналіз програмно-апаратних засобів для розробки модуля

Для реалізації IDS-модуля було підібрано апаратні та програмні засоби, які відповідають вимогам до моніторингу локальної мережі. Це дозволяє забезпечити стабільну роботу системи в реальному часі, мінімізувати затримки обробки мережі та забезпечити точну класифікацію подій. В якості апаратного забезпечення було обрано мікрокомп’ютер Raspberry Pi 4B (рисунок 3.1).

Рисунок 3.1 – Модуль розробки Raspberry Pi 4B

Raspberry Pi 4B обрано через наявність гігабітного Ethernet-порту, підтримку Linux і достатню продуктивність для роботи IDS-сенсора в локальній мережі. Додатковою перевагою є підтримка операційних систем на базі Linux. Це дає можливість встановлювати аналізатори трафіку, обробляти події, записувати їх в базу даних і відображати результати у веб-інтерфейсі адміністратора.

Для програмної розробки було обрано Suricata як головний інструмент для сигнатурного аналізу і створення звітів у форматі JSON (рисунок 3.2).

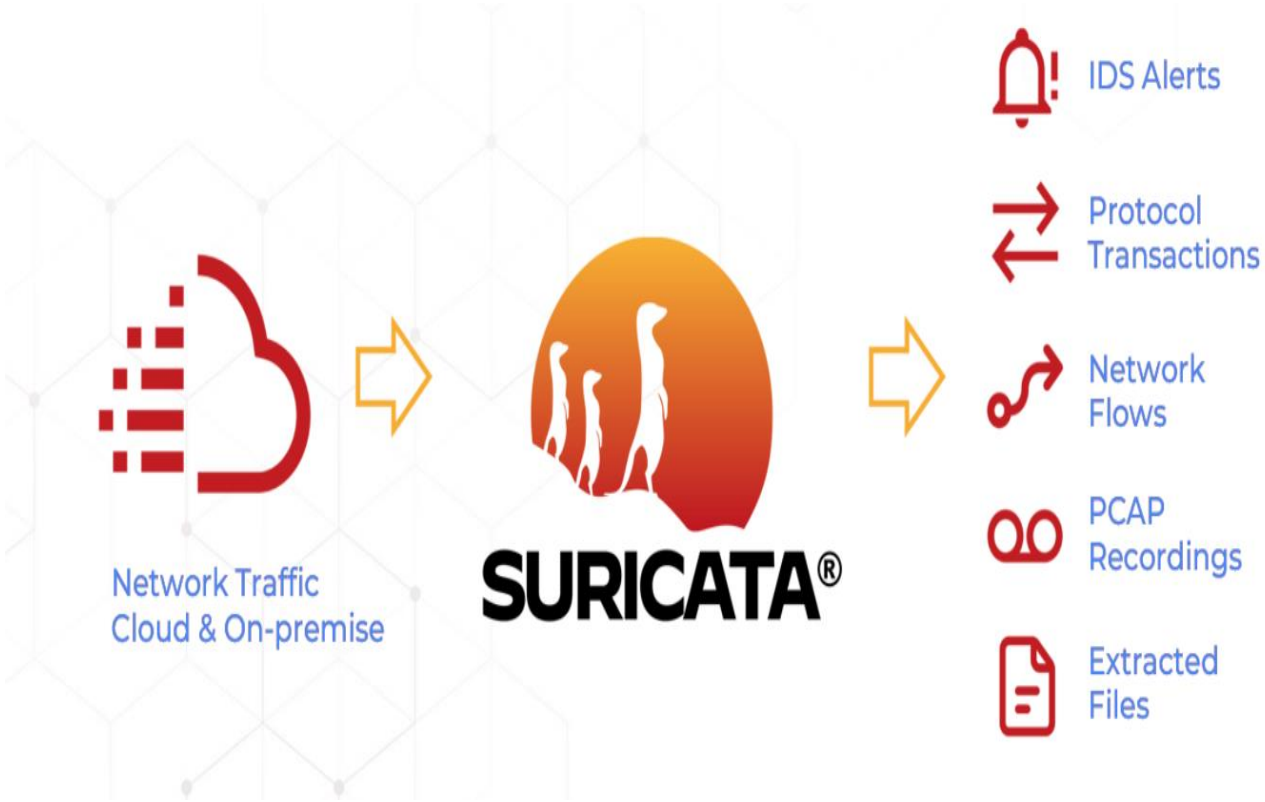


Рисунок 3.2 – Принцип роботи Suricata

Для зчитування даних з файлів у розробленій системі використовується скрипт, написаний мовою Python, який виконує попередню обробку даних і передає підготовлені ознаки на вхід класифікатора. В цій архітектурі Python використовується як проміжний рівень між Suricata, базою даних і модулем машинного навчання. Для реалізації ML-компонента використано бібліотеки TensorFlow і Keras, які дають змогу будувати та навчати нейронні мережі для обчислення показника аномальності на основі статистичних характеристик мережевих з'єднань. Завдяки цьому система не обмежується лише сигнатурами, а також оцінює статистичні ознаки мережевої активності.

На етапі розробки та тестування програмного модуля було використано середовище віртуалізації Oracle VM VirtualBox, в якому розгорнуто дистрибутив Debian (рисунок 3.3). Віртуальне середовище використано для безпечного тестування тому що в ньому можна моделювати атаки, перевіряти роботу компонентів і не впливати на реальну мережу. Крім того, це спрощує відтворення експериментів, налаштування мережевих сценаріїв та оцінювання працездатності розробленої моделі в контрольованих умовах.

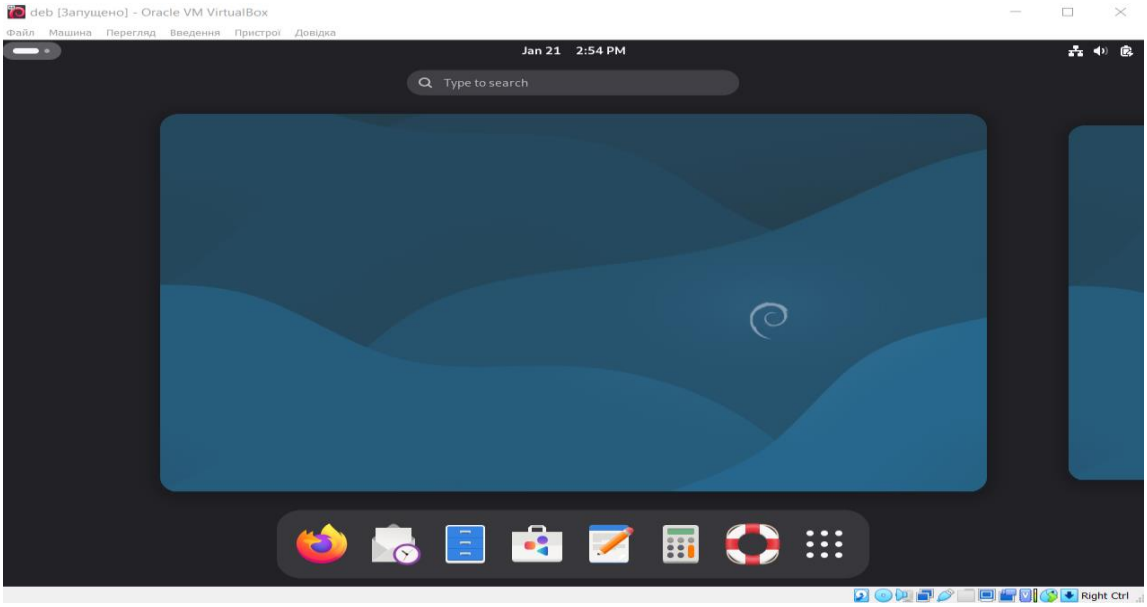


Рисунок 3.3 - Дистрибутив Debian

Така структура залишає можливість додавати нові методи аналізу та інтегрувати модуль в реальну локальну мережу.

3.2 Програмно-технологічне забезпечення модуля

Реалізацію IDS-модуля виконано поетапно. Окремо налаштовано захоплення трафіку, сигнатурний аналіз, обробку журналів, ML-класифікацію, базу даних і веб-інтерфейс. Дані проходять послідовний шлях від захоплення трафіку, попередньої обробки, сигнатурної перевірки, поведінкової оцінки, збереження результатів і відображення у веб-інтерфейсі. Така структура є зручною з боку тестування, тому що дозволяє перевіряти працездатність окремих функціональних блоків, адаптувати систему до різних умов мережевого середовища та виконувати вдосконалення окремих компонентів без необхідності повної переробки всієї програмної реалізації.

На початковому етапі виконується встановлення та базове налаштування програмного засобу Suricata, який є основою сигнатурного аналізу в розроблюваному модулі IDS. Для цього в середовищі Debian використовується пакетний менеджер apt, за допомогою якого інсталюються Suricata та допоміжні компоненти, необхідні для коректної обробки та аналізу результатів роботи

системи. На цьому етапі створюється основа для подальшої конфігурації правил виявлення, запуску моніторингу мережевого трафіку та інтеграції з іншими модулями системи. Приклад процесу встановлення Suricata наведено на рисунку 3.4.

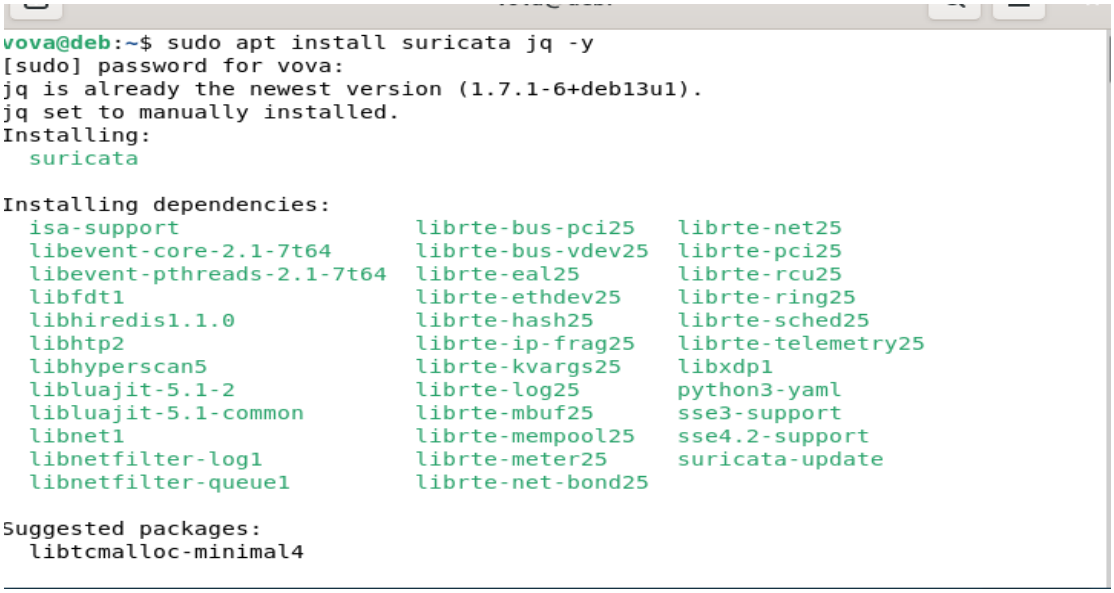


Рисунок 3.4 – Встановлення Suricata

Після завершення налаштування виконується перевірка працездатності системи, щоб переконатися у коректному запуску Suricata, правильності застосування правил та формуванні журналів подій (рисунок 3.5).

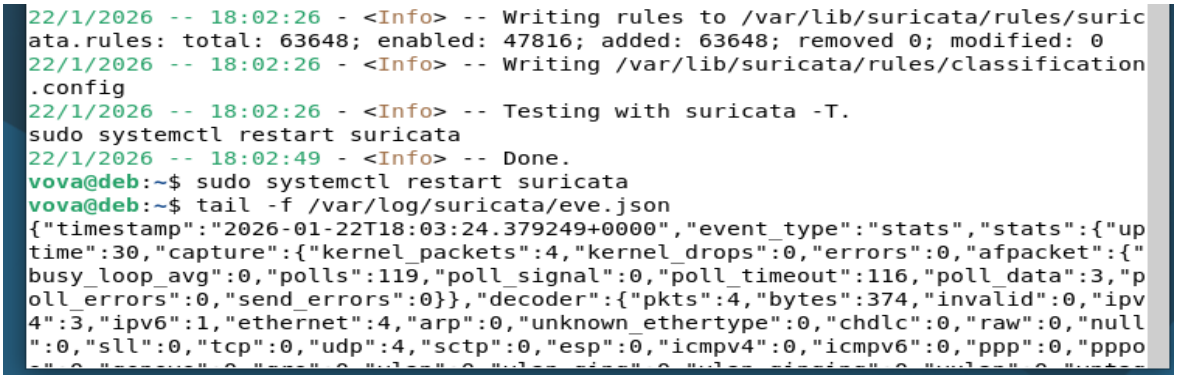


Рисунок 3.5 – Перевірка системи

Другий етап розробки пов’язаний із підготовкою середовища для ML-моделі та встановленням необхідних бібліотек у віртуальному середовищі (рисунок 3.6).

```
    Found existing installation: pip 25.1.1
    Uninstalling pip-25.1.1:
      Successfully uninstalled pip-25.1.1
Successfully installed pip-25.3
(venv) vova@deb:~/ids-ml$ python -m pip install numpy pandas tensorflow keras
Collecting numpy
  Downloading numpy-2.4.1-cp313-cp313-manylinux_2_27_x86_64.manylinux_2_28_x86_6
4.whl.metadata (6.6 kB)
Collecting pandas
  Downloading pandas-3.0.0-cp313-cp313-manylinux_2_24_x86_64.manylinux_2_28_x86_
64.whl.metadata (79 kB)
Collecting tensorflow
  Downloading tensorflow-2.20.0-cp313-cp313-manylinux_2_17_x86_64.manylinux2014_
```

Рисунок 3.6 – Встановлення бібліотек

Після встановлення необхідних бібліотек виконується створення програмного файла, який містить код для підготовки даних і подальшого навчання моделі машинного навчання (рисунок 3.7).

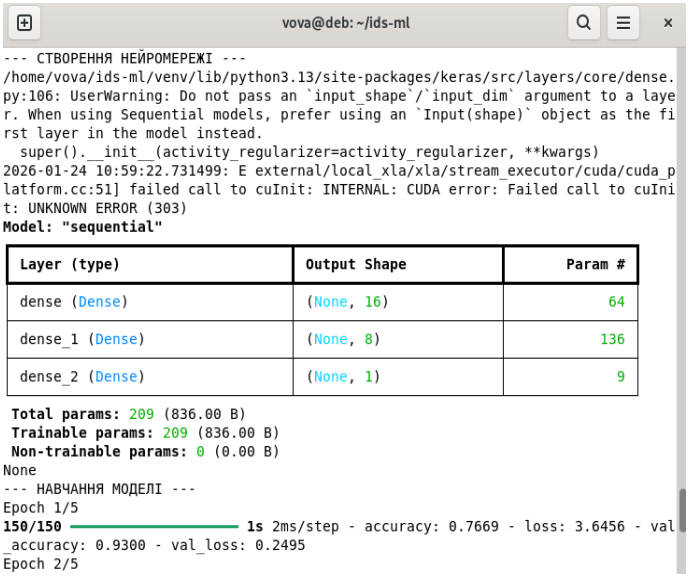
```
normal_data = np.random.normal(
    loc=[10, 500, 5],
    scale=[5, 100, 2],
    size=(3000, 3)
)
normal_labels = np.zeros(3000)


anomaly_data = np.random.normal(
    loc=[60, 10000, 100],
    scale=[10, 2000, 20],
    size=(3000, 3)
)
anomaly_labels = np.ones(3000)
```

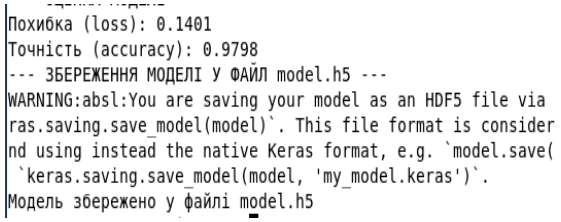
```
print("--- НАВЧАННЯ МОДЕЛІ ---")
history = model.fit(
    X, y,
    epochs=5,
    batch_size=32,
    verbose=1,
    validation_split=0.2
)
```

А - Підготовки ознак трафіку Б – Навчання моделі

Рисунок 3.7 – А (підготовки ознак трафіку), Б (Навчання моделі)

Після збереження коду проводиться перевірка запуску моделі, щоб підтвердити правильність її роботи, коректне проходження процесу навчання та відсутність помилок в програмній реалізації (рисунок 3.8).

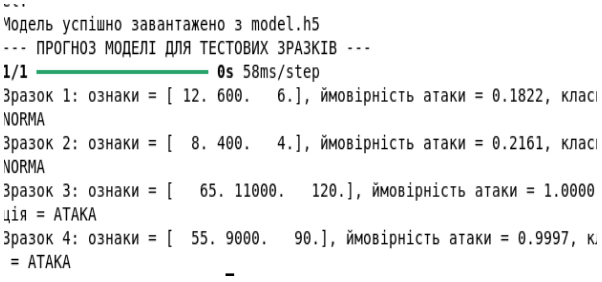




а) б)

Рисунок 3.8 – а) структура нейронної мережі та процес ініціалізації навчання, б) результати оцінки точності моделі та успішне збереження

Після перевірки працездатності моделі виконується підготовка окремого програмного коду для тестування її роботи на нових вхідних даних.



а) б)

Рисунок 3.9 – а) код тестування ML, б) Результат тестування

Після налаштування основних компонентів створено базу даних для збереження подій, які формуються під час аналізу мережевого трафіку (рисунок 3.10).

```
import os
import json
import sqlite3
import numpy as np
from tensorflow.keras.models import load_model

EVE_PATH = "/var/log/suricata/eve.json"
DB_PATH = "ids_events.db"
MODEL_PATH = "model.h5"


def init_db():
    conn = sqlite3.connect(DB_PATH)
    cur = conn.cursor()
    cur.execute("""
    CREATE TABLE IF NOT EXISTS events (
        id INTEGER PRIMARY KEY AUTOINCREMENT,
        timestamp TEXT,
        src_ip TEXT,
        src_port INTEGER,
```

Рисунок 3.10 - База даних

Після налаштування Suricata та навчання моделі реалізовано модуль, який об’єднує результати сигнатурного і поведінкового аналізу. В межах його роботи здійснюється зчитування системних журналів Suricata, обробка та виділення статистичних метрик, а також їх класифікація з використанням нейронної мережі. Відповідний приклад зчитування і парсингу журналів показано на рисунку 3.11.

```
with open(EVE_PATH, "r") as f:
    all_lines = f.readlines()
    lines = all_lines[-N_LINES:]

stats_events = []

for line in lines:
    line = line.strip()
    if not line:
        continue
    try:
        obj = json.loads(line)
    except json.JSONDecodeError:
        continue

    if obj.get("event_type") == "stats":
        stats_events.append(obj)
```

Рисунок 3.11 - Зчитування та парсинг журналів

Для того щоб нейронна мережа могла працювати з журналами Suricata, записи попередньо перетворюються на числові ознаки. Для цього реалізовано алгоритм отримання параметрів трафіку. На рисунку 3.12 наведено програмну реалізацію екстракції ознак в реальному часі.

```
uptime = float(stats.get("uptime", 0.0))

decoder = stats.get("decoder", {})
pkts = float(decoder.get("pkts", 0.0))
bytes_total = float(decoder.get("bytes", 0.0))

if uptime > 0:
    freq = pkts / uptime
else:
    freq = 0.0

dur_minutes = uptime / 60.0
```

Рисунок 3.12 - Програмна реалізація екстракції ознак у реальному часі

На останньому етапі створено веб-застосунок, який об’єднує збір, обробку та відображення результатів аналізу. Його реалізовано на мові Python із використанням Flask, що дозволяє поєднати серверну логіку з відображенням результатів роботи IDS-модуля. На рисунку 3.13 наведено фрагмент підключення бібліотек і налаштування середовища.

```
import os
import json
import sqlite3
import numpy as np
from flask import Flask, render_template_string
from tensorflow.keras.models import load_model

EVE_PATH = "/var/log/suricata/eve.json"
MODEL_PATH = "model.h5"
DB_PATH = "ids_events.db"

app = Flask(__name__)

TEMPLATE = """
```

Рисунок 3.13 - Ініціалізація системи та завантаження ресурсів

Далі реалізується модуль отримання статистичних даних від Suricata, основою якого є функція get_last_stats, призначена для вибірки та обробки останніх записів із системного журналу (рисунок 3.14).

```python
def get_last_stats():
    if not os.path.exists(EVE_PATH):
        return None
    with open(EVE_PATH, "r") as f:
        lines = f.readlines()
    lines = lines[-100:]
    stats_events = []
    for line in lines:
        line = line.strip()
        if not line:
            continue
        try:
            obj = json.loads(line)
        except json.JSONDecodeError:
            continue
        if obj.get("event_type") == "stats":
            stats_events.append(obj)
    if not stats_events:
        return None
    return stats_events[-1]
```

Рисунок 3.14 - Алгоритм зчитування та фільтрації логів

Ця функція отримує останні записи з журналу та використовує їх для оновлення показників на веб-сторінці. Для підвищення ефективності обробки реалізовано оптимізоване зчитування файлу eve.json, за якого аналізується не весь журнал подій, а лише його завершальна частина, тобто останні 100 рядків. Це зменшує навантаження на систему, тому що для оновлення інтерфейсу не потрібно кожен раз обробляти весь журнал.

Рисунок 3.15 ілюструє основну логіку системи, включаючи формування ознак на основі мережевих даних та їх подальшу класифікацію.

```python
stats = last_stats.get("stats", {})

uptime = float(stats.get("uptime", 0.0))
decoder = stats.get("decoder", {})
pkts = float(decoder.get("pkts", 0.0))
bytes_total = float(decoder.get("bytes", 0.0))

if uptime > 0:
    freq = pkts / uptime
    bytes_per_sec = bytes_total / uptime
else:
    freq = 0.0
    bytes_per_sec = 0.0

dur_minutes = uptime / 60.0

freq_feature = freq
bytes_feature = bytes_per_sec / 100.0
dur_feature = 1.0
```

Рисунок 3.15 - Формування вектора ознак та класифікація

На цьому етапі виконується перетворення технічних метрик мережевої активності у вектор ознак, придатний для подальшої обробки моделлю машинного навчання. В процесі обробки система автоматично обчислює похідні показники, зокрема частоту пакетів freq та тривалість сесії dur_minutes, які використовуються як інформативні характеристики для аналізу. Сформований вектор ознак передається на вхід нейронної мережі, яка на основі отриманих даних обчислює ймовірність наявності атаки.

3.3 Тестування модуля

Після реалізації основних компонентів проведено тестування IDS-модуля. Інтерфейс системи відкривається у веб-браузері у вигляді простої веб-сторінки, яка містить два основні інформаційні блоки. На верхній частині сторінки відображаються агреговані параметри мережевого трафіку, такі як час роботи сенсора, кількість пакетів, обсяг переданих даних і середня частота пакетів. В нижній частині сторінки відображається результат поведінкового аналізу: статус «АТАКА» або «NORMA» і розрахована ймовірність атаки. Сторінка автоматично оновлюється з інтервалом 5 секунд, що забезпечує відображення поточного стану мережі в режимі, наближеному до реального часу.

Для перевірки працездатності моделі було обрано сценарій, за якого спочатку імітується атака на мережу (рисунок 3.16).

```
(venv) vova@deb:~/ids-ml$ ping -c 40 8.8.8.8
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.
64 bytes from 8.8.8.8: icmp_seq=1 ttl=118 time=25.9 ms
64 bytes from 8.8.8.8: icmp_seq=2 ttl=118 time=22.0 ms
```

Рисунок 3.16 – Імітація атаки на мережу

Сигнал атаки генерується вручну за допомогою команди ping, при цьому задається контрольована кількість пакетів, що надходять у систему, щоб уникнути

її перевантаження. Такий підхід дозволяє перевірити реакцію IDS на аномальну мережеву активність у контрольованих умовах.

Такий сценарій дав змогу перевірити, чи правильно система реагує на аномальну активність і чи змінює оцінку ймовірності атаки. Проведене тестування дозволило оцінити роботу системи як в умовах аномальної активності, так і за нормального стану мережі, що є важливим для зменшення кількості хибних спрацювань. Отримані дані передаються до веб-сервісу, де відображається статистична інформація та результат класифікації у вигляді стану атаки або нормального режиму роботи мережі. На рисунку 3.17 наведено веб-сервіс розробленого модуля.

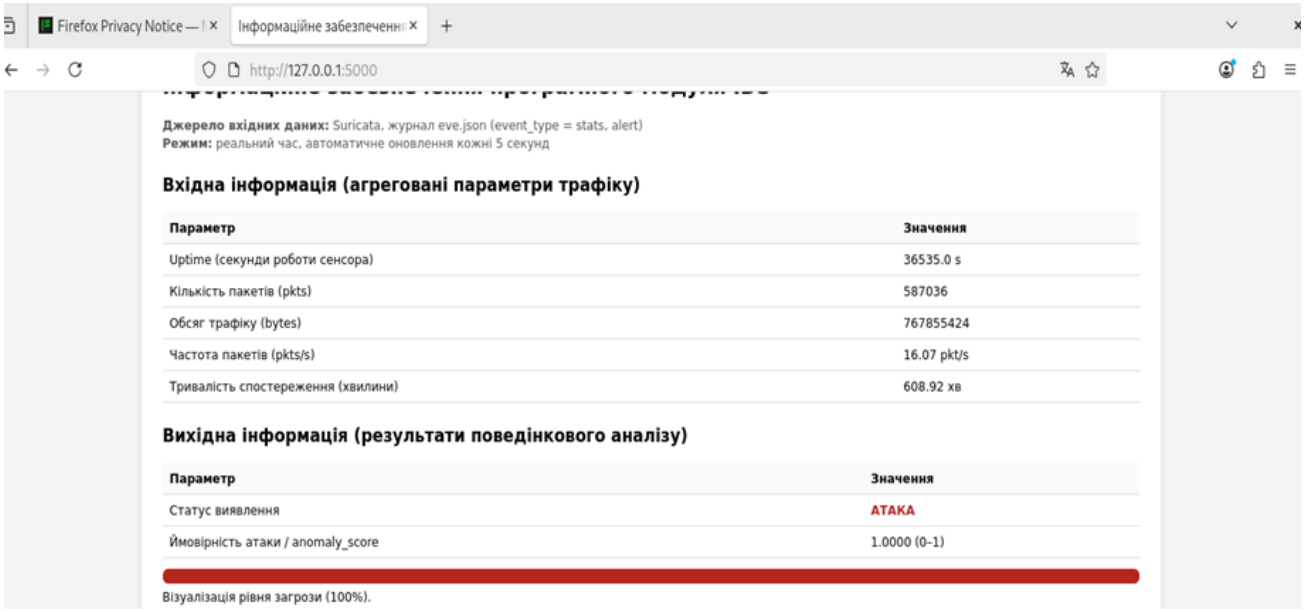


Рисунок 3.17 – Веб-сервіс системи

Веб-інтерфейс також містить блок перегляду останніх подій, зафіксованих IDS-модулем. Всі виявлені мережеві події, статистичні показники та результати класифікації спочатку зберігаються в раніше створеній базі даних, що забезпечує їх упорядковане накопичення та подальшу обробку. Після цього дані передаються до веб-застосунку, де відображаються у зручному для аналізу вигляді.

Завдяки цьому користувач може спостерігати останні зміни стану мережі, переглядати зафіксовані інциденти, оцінки аномальності, сигнатури спрацювань і класифікацію подій як нормального стану або потенційної атаки. Це дає

можливість контролювати поточну ситуацію в мережі та аналізувати історію спрацювань IDS у процесі тестування системи. На рисунку 3.18 зображено події, зафіксовані в мережі.

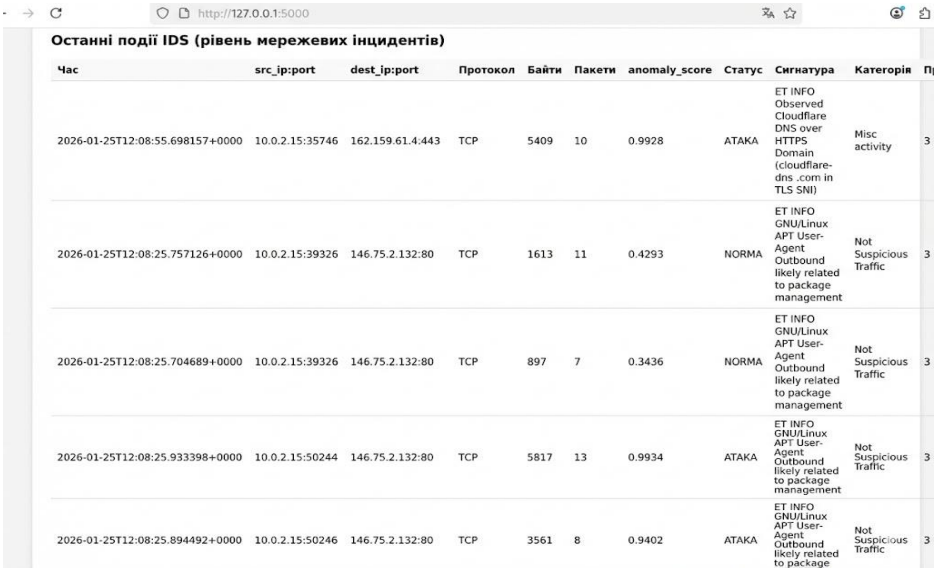


Рисунок 3.18 – Події в мережі

Після завершення імітації атаки система повернулася до нормального режиму та змінила статус на «NORMA». Водночас інтерфейс продовжував показувати рівень загрози та ймовірність атаки, що дає змогу відстежувати залишкові відхилення у трафіку. На рисунку 3.19 зображено роботу при нормальному стані мережі.

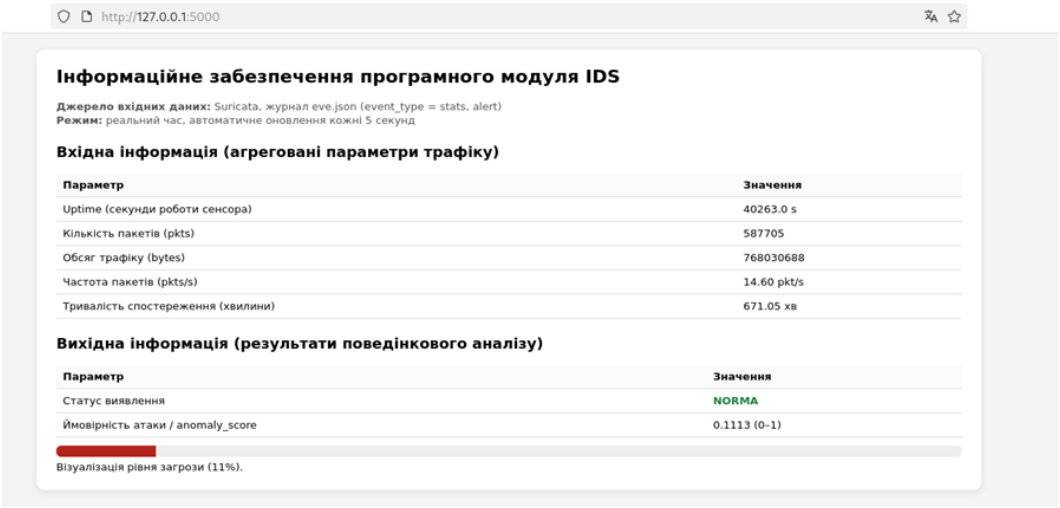


Рисунок 3.19 – Робота мережі в нормальному стані

Тестування показало, що розроблений IDS-модуль коректно працює в змодельованому мережевому середовищі. Під час перевірки модуль виявляв аномальну активність, формував спрацювання та змінював стан після повернення трафіку до нормального режиму.

Окремо було перевірено механізм запису подій до бази даних, що забезпечує їх подальше збереження, перегляд у майбутньому та використання для моніторингу в режимі, наближеному до реального часу. На рисунку 3.20 зображено базу даних з записами зафіксованих подій.

```
(venv) vova@deb:~/ids-ml$ sqlite3 ids_events.db
SQLite version 3.46.1 2024-08-13 09:16:08
Enter ".help" for usage hints.
sqlite> SELECT timestamp, src_ip, dest_ip, proto, bytes, pkts, anomaly_score, la
bel, signature, severity
FROM events
ORDER BY id DESC
LIMIT 10;
2026-01-25T12:08:52.698157+0000|10.0.2.15|162.159.61.4|TCP|5409|10|0.99276506900
7874|ATAKA|ET INFO Observed Cloudflare DNS over HTTPS Domain (cloudflare-dns .co
m in TLS SNI)|3
2026-01-25T12:08:52.698157+0000|10.0.2.15|162.159.61.4|TCP|5409|10|0.99276506900
7874|ATAKA|ET INFO Observed Cloudflare DNS over HTTPS Domain (cloudflare-dns .co
m in TLS SNI)|3
2026-01-25T12:08:25.757126+0000|10.0.2.15|146.75.2.132|TCP|1613|11|0.42930746078
4912|NORMA|ET INFO GNU/Linux APT User-Agent Outbound likely related to package m
anagement|3
2026-01-25T12:08:25.704689+0000|10.0.2.15|146.75.2.132|TCP|897|7|0.3436314761638
64|NORMA|ET INFO GNU/Linux APT User-Agent Outbound likely related to package man
agement|3
2026-01-25T12:08:25.701868+0000|10.0.2.15|146.75.2.132|TCP|960|7|0.3628056645393
37|NORMA|ET INFO GNU/Linux APT User-Agent Outbound likely related to package man
agement|3
```

Рисунок 3.20 – Події в базі даних

Результати тестування показують, що систему можна розвивати далі, тобто розширювати набір ознак, уточнювати правила та вдосконалювати механізми реагування. Завдяки наявності веб-інтерфейсу адміністратор може швидко переглядати збережені логи, виконувати аналіз інцидентів, відокремлювати найбільш критичні події та своєчасно вживати заходів для протидії мережевим атакам.

ВИСНОВКИ

Під час виконання кваліфікаційної роботи було розроблено програмний модуль IDS для локальної мережі, який поєднує сигнатурний та поведінковий аналіз мережевого трафіку з використанням засобів машинного навчання. Розроблена система забезпечує виявлення підозрілої активності, оцінювання ймовірності атаки, збереження подій у базі даних та відображення результатів у веб-інтерфейсі в режимі, наближеному до реального часу.

1. Проведено аналіз предметної області систем виявлення та запобігання вторгненням в локальних мережах. Розглянуто поняття IDS/IPS, класифікацію таких систем, основні моделі загроз, а також сигнатурний, поведінковий і політико-орієнтований підходи до виявлення вторгнень.

2. Виконано огляд відкритих програмних платформ для побудови IDS/IPS, зокрема Suricata, Snort і Zeek. Порівняння цих платформ дало змогу обґрунтувати використання Suricata як основного засобу сигнатурного аналізу.

3. Сформульовано постановку задачі, для IDS-модуля для локальної мережі з комбінованим аналізом трафіку, можливістю виявлення відомих загроз за сигнатурами, фіксацією аномальної поведінки, накопиченням історії подій та відображенням результатів у зручному для адміністратора вигляді.

4. Розроблено функціональну структуру системи та алгоритмічне забезпечення модуля. Описано підсистеми моніторингу трафіку, декодування і розбору протоколів, сигнатурного аналізу, кореляції подій, поведінкового виявлення аномалій, а також формування звітів і оповіщень у реальному часі. Сформовано сценарії використання системи для виявлення сигнатурних атак, поведінкових відхилень і перегляду журналу подій адміністратором.

5. Розроблено інформаційне забезпечення програмного модуля у вигляді структури вхідних і вихідних даних, а також логічної моделі бази даних для збереження мережевих подій, статистичних показників, сигнатур і результатів класифікації. Це забезпечує накопичення історії інцидентів, подальший аналіз та зручний доступ до зафіксованих подій.

6. Обґрунтовано вибір програмно-апаратних засобів для реалізації системи. Для побудови модуля використано Raspberry Pi 4B як платформу розгортання, Suricata для сигнатурного аналізу, Python як інтеграційний рівень між компонентами системи, TensorFlow і Keras для реалізації поведінкового аналізу, SQLite для збереження подій та Flask для створення веб-інтерфейсу.

7. Реалізовано програмно-технологічне забезпечення системи та проведено тестування її роботи. Під час тестування перевірено встановлення і налаштування Suricata, навчання та запуск ML-моделі, зчитування і парсинг журналів eve.json, екстракцію ознак, класифікацію подій, запис результатів до бази даних та їх відображення у веб-застосунку.

Розроблений IDS-модуль можна використати як основу для подальшого розвитку системи моніторингу локальної мережі.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. ENISA Threat Landscape 2024. Heraklion : European Union Agency for Cybersecurity, 2024. URL: https://www.enisa.europa.eu/publications/enisa-threat-landscape-2024 (дата звернення: 05.05.2026).

2. Denning D. E. An Intrusion-Detection Model. IEEE Transactions on Software Engineering. 1987. Vol. SE-13, no. 2. P. 222–232. URL: https://www.cs.colostate.edu/~cs656/reading/ieee-se-13-2.pdf (дата звернення: 05.05.2026).

3. Anderson J. P. Computer Security Threat Monitoring and Surveillance. Fort Washington, PA : James P. Anderson Co., 1980. URL: https://seclab.cs.ucdavis.edu/projects/history/papers/ande80.pdf (дата звернення: 05.05.2026).

4. Heberlein L. T., Dias G. V., Levitt K. N., Mukherjee B., Wood J., Wolber D. A Network Security Monitor. Proceedings of the 1990 IEEE Symposium on Research in Security and Privacy. 1990. P. 296–304. URL: https://seclab.cs.ucdavis.edu/papers/pdfs/th-gd-90.pdf (дата звернення: 05.05.2026).

5. Scarfone K., Mell P. Guide to Intrusion Detection and Prevention Systems (IDPS). NIST Special Publication 800-94. Gaithersburg : National Institute of Standards and Technology, 2007. URL: https://nvlpubs.nist.gov/nistpubs/legacy/sp/nistspecialpublication800-94.pdf (дата звернення: 05.05.2026).

6. Diana L., Dini P., Paolini D. Overview on Intrusion Detection Systems for Computers Networking Security. Computers. 2025. Vol. 14, no. 3. Art. 87. URL: https://www.mdpi.com/2073-431X/14/3/87 (дата звернення: 05.05.2026).

7. Hozouri A., Mirzaei A., Effatparvar M. A comprehensive survey on intrusion detection systems with advances in machine learning, deep learning and emerging cybersecurity challenges. Discover Artificial Intelligence. 2025. Vol. 5. Art. 314. DOI: https://doi.org/10.1007/s44163-025-00578-1.

8. Rahman M. M., Shakil S. A., Mustakim M. R. A survey on intrusion detection system in IoT networks. Cyber Security and Applications. 2025. Vol. 3. Art. 100082. DOI: https://doi.org/10.1016/j.csa.2024.100082.

9. Li Y., Li Z., Li M. A comprehensive survey on intrusion detection algorithms. Computers and Electrical Engineering. 2025. Vol. 121. Art. 109863. DOI: https://doi.org/10.1016/j.compeleceng.2024.109863.

10. Zhang Y., Muniyandi R. C., Qamar F. A Review of Deep Learning Applications in Intrusion Detection Systems: Overcoming Challenges in Spatiotemporal Feature Extraction and Data Imbalance. Applied Sciences. 2025. Vol. 15, no. 3. Art. 1552. DOI: https://doi.org/10.3390/app15031552.

11. Shyaa M. A., Ibrahim N. F., Zainol Z., Abdullah R., Anbar M., Alzubaidi L. Evolving cybersecurity frontiers: A comprehensive survey on concept drift and feature dynamics aware machine and deep learning in intrusion detection systems. Engineering Applications of Artificial Intelligence. 2024. Vol. 137. Art. 109143. DOI: https://doi.org/10.1016/j.engappai.2024.109143.

12. Zhong M., Lin M., Zhang C., Xu Z. A survey on graph neural networks for intrusion detection systems: Methods, trends and challenges. Computers & Security. 2024. Vol. 141. Art. 103821. DOI: https://doi.org/10.1016/j.cose.2024.103821.

13. Okolie S. A., Amadi C. A., Odii J. N., Nwokorie E. C., Onyemauche U. C. Anomaly detection in heterogeneous cybersecurity data. Franklin Open. 2025. Vol. 13. Art. 100426. DOI: https://doi.org/10.1016/j.fraope.2025.100426.

14. Holdbrook R., Odeyomi O., Yi S., Roy K. Network-Based Intrusion Detection for Industrial and Robotics Systems: A Comprehensive Survey. Electronics. 2024. Vol. 13, no. 22. Art. 4440. URL: https://www.mdpi.com/2079-9292/13/22/4440 (дата звернення: 05.05.2026).

15. Satılmış H., Akleylek S., Tok Z. Y. A Systematic Literature Review on Host-Based Intrusion Detection Systems. IEEE Access. 2024. Vol. 12. P. 27237–27266. DOI: https://doi.org/10.1109/ACCESS.2024.3367004.

16. Kwon H.-Y., Kim T., Lee M.-K. Advanced Intrusion Detection Combining Signature-Based and Behavior-Based Detection Methods. Electronics. 2022. Vol. 11, no. 6. Art. 867. DOI: https://doi.org/10.3390/electronics11060867.

17. Otoum Y., Nayak A. AS-IDS: Anomaly and Signature Based IDS for the Internet of Things. Journal of Network and Systems Management. 2021. Vol. 29, no. 3. Art. 23. DOI: https://doi.org/10.1007/s10922-021-09589-6.

18. Ahmad Z., Khan A. S., Shiang C. W., Abdullah J., Ahmad F. Network intrusion detection system: A systematic study of machine learning and deep learning approaches. Transactions on Emerging Telecommunications Technologies. 2021. Vol. 32, no. 1. Art. e4150. DOI: https://doi.org/10.1002/ett.4150.

19. Khraisat A., Alazab A. A critical review of intrusion detection systems in the internet of things: techniques, deployment strategy, validation strategy, attacks, public datasets and challenges. Cybersecurity. 2021. Vol. 4. Art. 18. DOI: https://doi.org/10.1186/s42400-021-00077-7.

20. Panigrahi R., Borah S., Bhoi A. K., Ijaz M. F., Pramanik M., Jhaveri R. H., Chowdhary C. L. Performance assessment of supervised classifiers for designing intrusion detection systems: A comprehensive review and recommendations for future research. Mathematics. 2021. Vol. 9, no. 6. Art. 690. DOI: https://doi.org/10.3390/math9060690.

21. Arshad J., Azad M. A., Amad R., Salah K., Alazab M., Iqbal R. A Review of Performance, Energy and Privacy of Intrusion Detection Systems for IoT. Electronics. 2020. Vol. 9, no. 4. Art. 629. DOI: https://doi.org/10.3390/electronics9040629.

22. Agrawal S., Sarkar S., Aouedi O., Yenduri G., Piamrat K., Alazab M., Bhattacharya S., Maddikunta P. K. R., Gadekallu T. R. Federated Learning for intrusion detection system: Concepts, challenges and future directions. Computer Communications. 2022. Vol. 195. P. 346–361.

23. Al Jallad K., Aljnidi M., Desouki M. S. Anomaly detection optimization using big data and deep learning to reduce false-positive. Journal of Big Data. 2020. Vol. 7. Art. 68.

24. Saranya T., Sridevi S., Deisy C., Chung T. D., Khan M. Performance Analysis of Machine Learning Algorithms in Intrusion Detection System: A Review. Procedia Computer Science. 2020. Vol. 171. P. 1251–1260.

25. Dini P., Elhanashi A., Begni A., Saponara S., Zheng Q., Gasmi K. Overview on Intrusion Detection Systems Design Exploiting Machine Learning for Networking Cybersecurity. Applied Sciences. 2023. Vol. 13, no. 13. Art. 7507. DOI: https://doi.org/10.3390/app13137507.

26. Roesch M. Snort – Lightweight Intrusion Detection for Networks. Proceedings of the 13th USENIX Conference on System Administration (LISA'99). Seattle, 1999. P. 229–238. URL: https://www.usenix.org/event/lisa99/full_papers/roesch/roesch.pdf (дата звернення: 05.05.2026).

27. Paxson V. Bro: A System for Detecting Network Intruders in Real-Time. Computer Networks. 1999. Vol. 31, no. 23–24. P. 2435–2463. URL: https://www.usenix.org/publications/library/proceedings/sec98/full_papers/paxson/paxson.pdf (дата звернення: 05.05.2026).

28. What is Suricata. Suricata User Guide. URL: https://docs.suricata.io/en/latest/what-is-suricata.html (дата звернення: 05.05.2026).

29. About Zeek. Book of Zeek. URL: https://docs.zeek.org/en/current/about.html (дата звернення: 05.05.2026).

30. Raspberry Pi 4 Model B Datasheet. Raspberry Pi Ltd. 2024. URL: https://pip.raspberrypi.com/documents/RP-008341-DS-raspberry-pi-4-datasheet.pdf (дата звернення: 05.05.2026).

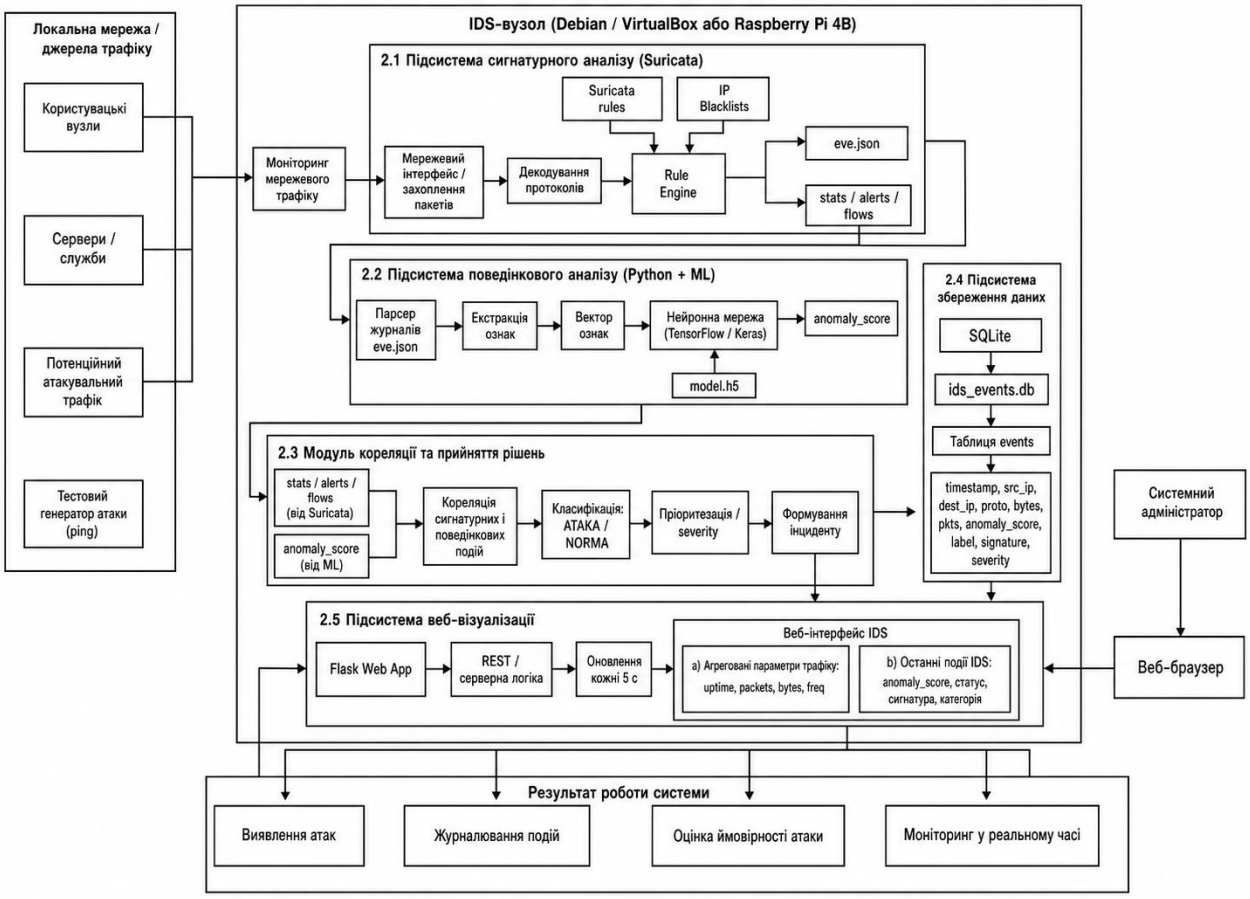
31. Комар М.П., Саченко А.О., Васильків Н.М., Гладій Г.М., Коваль В.С., Ліп'яніна-Гончаренко Х.В. Методичні рекомендації до виконання кваліфікаційної роботи з освітньо-професійної програми «Комп'ютерні науки» спеціальності 122 «Комп'ютерні науки» за першим (бакалаврським) рівнем вищої освіти. Тернопіль: ЗУНУ, 2024. 52 с.

32. Василічишена В.С. Модуль IDS для локальної мережі з комбінованим сигнатурним і поведінковим аналізом трафіку // Матеріали X Міжнародної

студентської наукової конференції «Діджиталізація науки як виклик сьогодення». Миколаїв, 2026. С.127-129.

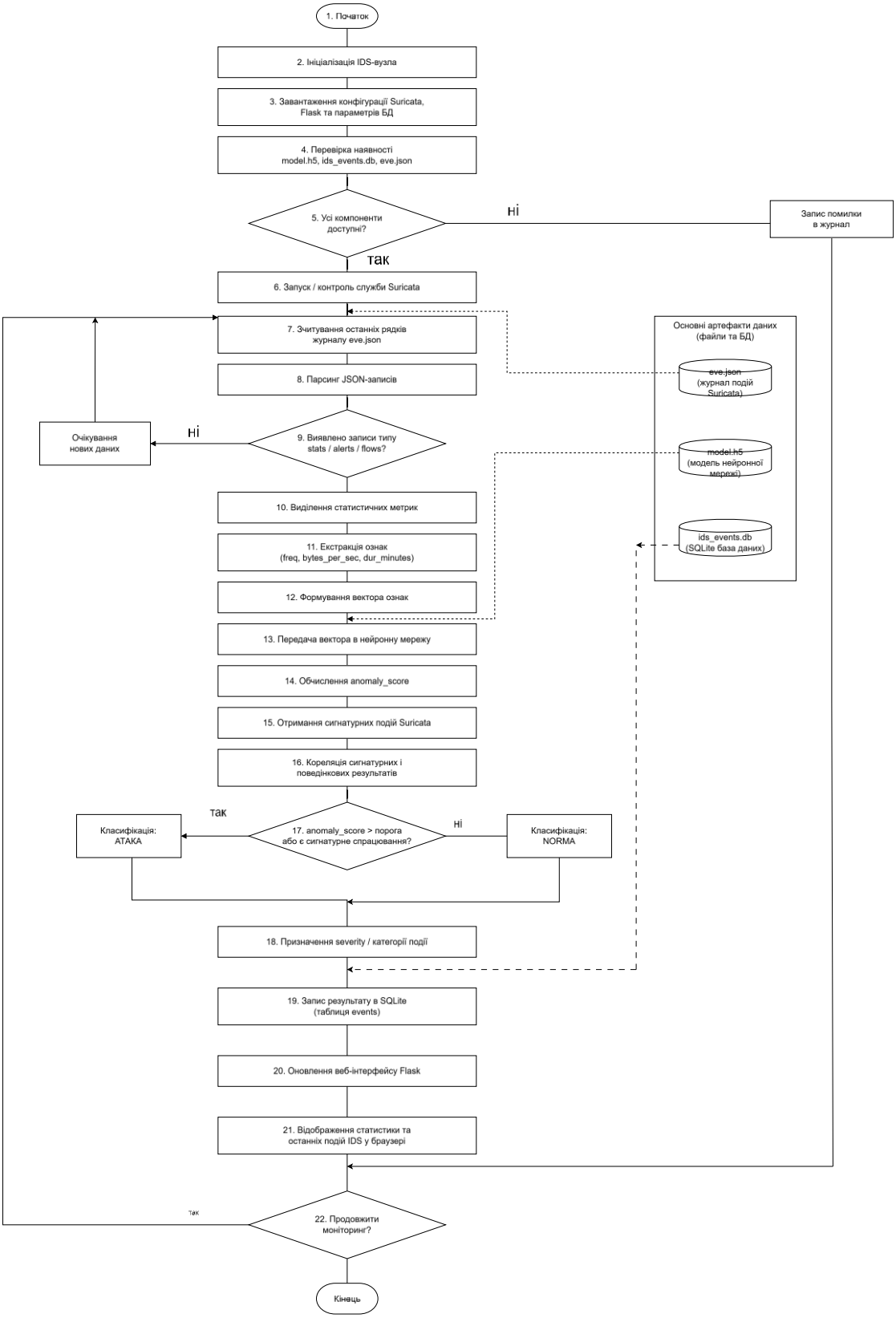
Додаток А

Загальна архітектура модуля IDS



Додаток Б

Схема функціонування модуля IDS



Додаток В

Програмний код основних субмодулів

```python
import numpy as np
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense

print("--- ГЕНЕРАЦІЯ ДАНИХ ---")

# Генеруємо "нормальний" трафік
normal_data = np.random.normal(
    loc=[10, 500, 5],
    scale=[5, 100, 2],
    size=(3000, 3)
)
normal_labels = np.zeros(3000)  # мітка 0 = Норма

# Генеруємо "аномальний" трафік
anomaly_data = np.random.normal(
    loc=[60, 10000, 100],
    scale=[10, 2000, 20],
    size=(3000, 3)
)
anomaly_labels = np.ones(3000)  # мітка 1 = Атака

# Об'єднуємо дані
X = np.vstack((normal_data, anomaly_data))
y = np.hstack((normal_labels, anomaly_labels))

# Перемішуємо дані
indices = np.arange(X.shape[0])
np.random.shuffle(indices)
X = X[indices]
y = y[indices]

print("Форма X:", X.shape)
print("Форма y:", y.shape)

print("--- СТВОРЕННЯ НЕЙРОМЕРЕЖІ ---")

model = Sequential([
    Dense(16, input_dim=3, activation='relu'),
    Dense(8, activation='relu'),
    Dense(1, activation='sigmoid')
])
```

```python
model.compile(
    loss='binary_crossentropy',
    optimizer='adam',
    metrics=['accuracy']
)

print(model.summary())

print("--- НАВЧАННЯ МОДЕЛІ ---")
history = model.fit(
    X, y,
    epochs=5,
    batch_size=32,
    verbose=1,
    validation_split=0.2
)

print("--- ОЦІНКА МОДЕЛІ ---")
loss, acc = model.evaluate(X, y, verbose=0)
print(f"Похибка (loss): {loss:.4f}")
print(f"Точність (accuracy): {acc:.4f}")

print("--- ЗБЕРЕЖЕННЯ МОДЕЛІ У ФАЙЛ model.h5 ---")
model.save("model.h5")
print("Модель збережено у файлі model.h5")

------------------  код для ML


(для заходу
 cd ids_project
source ids_env/bin/activate) то перед кодом




-----------------------------------------------
тестування ML

import numpy as np
from tensorflow.keras.models import load_model

print("--- ЗАВАНТАЖЕННЯ МОДЕЛІ ---")
```

```python
model = load_model("model.h5")
print("Модель успішно завантажено з model.h5")


normal_samples = np.array([
    [12, 600, 6],   # схоже на "норму"
    [8,  400, 4]
], dtype=float)

attack_samples = np.array([
    [65, 11000, 120],  # схоже на "атаку"
    [55, 9000,  90]
], dtype=float)

test_X = np.vstack((normal_samples, attack_samples))

print("--- ПРОГНОЗ МОДЕЛІ ДЛЯ ТЕСТОВИХ ЗРАЗКІВ ---")
pred_probs = model.predict(test_X)

for i, x in enumerate(test_X):
    prob = pred_probs[i][0]
    label = 1 if prob >= 0.5 else 0
    label_str = "АТАКА" if label == 1 else "НОРМА"
    print(f"Зразок {i+1}: ознаки = {x}, ймовірність атаки = {prob:.4f}, класифікація = {label_str}")



-----------------------------------------------------------------

ВД



import os
import json
import sqlite3
import numpy as np
from tensorflow.keras.models import load_model

EVE_PATH = "/var/log/suricata/eve.json"
DB_PATH = "ids_events.db"
MODEL_PATH = "model.h5"
```

```python
def init_db():
    conn = sqlite3.connect(DB_PATH)
    cur = conn.cursor()
    cur.execute("""
    CREATE TABLE IF NOT EXISTS events (
        id INTEGER PRIMARY KEY AUTOINCREMENT,
        timestamp TEXT,
        src_ip TEXT,
        src_port INTEGER,
        dest_ip TEXT,
        dest_port INTEGER,
        proto TEXT,
        pkts INTEGER,
        bytes INTEGER,
        anomaly_score REAL,
        label TEXT,
        signature_id INTEGER,
        signature TEXT,
        category TEXT,
        severity INTEGER
    )
    """)
    conn.commit()
    return conn


def process_eve(conn, model, max_lines=1000):
    if not os.path.exists(EVE_PATH):
        print(f"Файл {EVE_PATH} не знайдено")
        return

    with open(EVE_PATH, "r") as f:
        lines = f.readlines()

    lines = lines[-max_lines:]
    cur = conn.cursor()
    inserted = 0

    for line in lines:
        line = line.strip()
        if not line:
            continue
        try:
            obj = json.loads(line)
```

```
        except json.JSONDecodeError:
            continue

        if obj.get("event_type") != "alert":
            continue

        ts = obj.get("timestamp")
        src_ip = obj.get("src_ip")
        src_port = obj.get("src_port")
        dest_ip = obj.get("dest_ip")
        dest_port = obj.get("dest_port")
        proto = obj.get("proto")

        alert = obj.get("alert", {}) or {}
        sig_id = alert.get("signature_id")
        signature = alert.get("signature")
        category = alert.get("category")
        severity = alert.get("severity")

        flow = obj.get("flow", {}) or {}
        pkts_ts = flow.get("pkts_toserver", 0) or 0
        pkts_tc = flow.get("pkts_toclient", 0) or 0
        bytes_ts = flow.get("bytes_toserver", 0) or 0
        bytes_tc = flow.get("bytes_toclient", 0) or 0
        pkts = int(pkts_ts + pkts_tc)
        bytes_total = int(bytes_ts + bytes_tc)

        dur_minutes = 1.0
        freq = pkts / dur_minutes if dur_minutes > 0 else float(pkts)

        feature_vector = np.array([[freq, float(bytes_total), dur_minutes]], dtype=float)
        prob = float(model.predict(feature_vector, verbose=0)[0][0])
        label = "АТАКА" if prob >= 0.5 else "NORMA"

        cur.execute("""
        INSERT INTO events
        (timestamp, src_ip, src_port, dest_ip, dest_port, proto,
         pkts, bytes, anomaly_score, label,
         signature_id, signature, category, severity)
        VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?)
        """, (
            ts, src_ip, src_port, dest_ip, dest_port, proto,
            pkts, bytes_total, prob, label,
            sig_id, signature, category, severity
```

```
        ))
        inserted += 1

    conn.commit()
    print(f"Додано {inserted} подій до бази даних.")


def main():
    print("Завантаження моделі...")
    model = load_model(MODEL_PATH)
    conn = init_db()
    process_eve(conn, model, max_lines=1000)
    conn.close()


if __name__ == "__main__":
    main()



---------------------------------------------

модуля комбінованого аналізу мережевого трафіку


import json
import numpy as np
from tensorflow.keras.models import load_model

EVE_PATH = "/var/log/suricata/eve.json"

print("--- ЗАВАНТАЖЕННЯ МОДЕЛІ IDS ---")
model = load_model("model.h5")
print("Модель успішно завантажено з model.h5")

print(f"--- ЧИТАННЯ ЖУРНАЛУ РОДІЙ {EVE_PATH} ---")

N_LINES = 200

lines = []
with open(EVE_PATH, "r") as f:
    all_lines = f.readlines()
    lines = all_lines[-N_LINES:]
```

```
stats_events = []

for line in lines:
    line = line.strip()
    if not line:
        continue
    try:
        obj = json.loads(line)
    except json.JSONDecodeError:
        continue

    if obj.get("event_type") == "stats":
        stats_events.append(obj)

if not stats_events:
    print("Немає подій типу 'stats' у останніх рядках eve.json.")
    exit(0)

last_stats = stats_events[-1]
timestamp = last_stats.get("timestamp")
stats = last_stats.get("stats", {})
uptime = float(stats.get("uptime", 0.0))

decoder = stats.get("decoder", {})
pkts = float(decoder.get("pkts", 0.0))
bytes_total = float(decoder.get("bytes", 0.0))

if uptime > 0:
    freq = pkts / uptime
else:
    freq = 0.0

dur_minutes = uptime / 60.0

feature_vector = np.array([[freq, bytes_total, dur_minutes]], dtype=float)

print("--- ОТРИМАНІ ОЗНАКИ З ЖУРНАЛУ SURICATA ---")
print(f"timestamp   = {timestamp}")
print(f"uptime (s)  = {uptime}")
print(f"pkts        = {pkts}")
print(f"bytes       = {bytes_total}")
print(f"freq (pkts/s) = {freq}")
print(f"dur (min)   = {dur_minutes}")
```

```
print("Вектор ознак для моделі:", feature_vector)

print("\n--- ПРОГНОЗ МОДЕЛІ ДЛЯ ПОТОЧНОГО СТАНУ ТРАФІКУ ---")
prob = model.predict(feature_vector, verbose=0)[0][0]
label = 1 if prob >= 0.5 else 0
label_str = "АТАКА" if label == 1 else "NORMA"

print(f"Ймовірність атаки = {prob:.4f}, класифікація = {label_str}")


---------------------------------------------------------------


веб сервіс

import os
import json
import sqlite3
import numpy as np
from flask import Flask, render_template_string
from tensorflow.keras.models import load_model

EVE_PATH = "/var/log/suricata/eve.json"
MODEL_PATH = "model.h5"
DB_PATH = "ids_events.db"

app = Flask(__name__)

TEMPLATE = """
<!doctype html>
<html lang="uk">
<head>
  <meta charset="utf-8">
  <title>Інформаційне забезпечення програмного модуля IDS</title>
  <meta http-equiv="refresh" content="5">
  <style>
    body {
      font-family: sans-serif;
      background: #f4f4f4;
      margin: 0;
      padding: 20px;
    }
    .card {
      max-width: 1100px;
```

```
      margin: 0 auto;
      background: #ffffff;
      border-radius: 10px;
      padding: 20px 24px;
      box-shadow: 0 2px 8px rgba(0,0,0,0.1);
    }
    h1 {
      margin-top: 0;
      font-size: 22px;
    }
    h2 {
      margin-top: 20px;
      font-size: 18px;
    }
    .status-ok {
      color: #1a7f37;
      font-weight: bold;
    }
    .status-bad {
      color: #b3261e;
      font-weight: bold;
    }
    table {
      width: 100%;
      border-collapse: collapse;
      margin-top: 8px;
    }
    th, td {
      border-bottom: 1px solid #ddd;
      padding: 6px 8px;
      text-align: left;
      font-size: 13px;
    }
    th {
      background: #fafafa;
    }
    .meta {
      font-size: 13px;
      color: #555;
      margin-bottom: 10px;
    }
    .prob-bar {
      margin-top: 10px;
      background: #eee;
```

```
      border-radius: 6px;
      overflow: hidden;
      height: 14px;
    }
    .prob-bar-inner {
      height: 100%;
      background: #b3261e;
    }
    .prob-value {
      font-size: 13px;
      margin-top: 4px;
    }
  </style>
</head>
<body>
  <div class="card">
    <h1>Інформаційне забезпечення програмного модуля IDS</h1>

    {% if error %}
      <p class="status-bad">{{ error }}</p>
    {% else %}
      <p class="meta">
        <strong>Джерело вхідних даних:</strong> Suricata, журнал eve.json (event_type = stats, alert)<br>
        <strong>Режим:</strong> реальний час, автоматичне оновлення кожні 5 секунд
      </p>

      <h2>Вхідна інформація (агреговані параметри трафіку)</h2>
      <table>
        <tr><th>Параметр</th><th>Значення</th></tr>
        <tr><td>Uptime (секунди роботи сенсора)</td><td>{{ uptime }} s</td></tr>
        <tr><td>Кількість пакетів (pkts)</td><td>{{ pkts }}</td></tr>
        <tr><td>Обсяг трафіку (bytes)</td><td>{{ bytes_total }}</td></tr>
        <tr><td>Частота пакетів (pkts/s)</td><td>{{ freq }} pkt/s</td></tr>
        <tr><td>Тривалість спостереження (хвилини)</td><td>{{ dur_minutes }} хв</td></tr>
      </table>

      <h2>Вихідна інформація (результати поведінкового аналізу)</h2>
      <table>
        <tr><th>Параметр</th><th>Значення</th></tr>
        <tr>
          <td>Статус виявлення</td>
          <td>
            {% if label == "АТАКА" %}
              <span class="status-bad">{{ label }}</span>
```

```
          {% else %}
            <span class="status-ok">{{ label }}</span>
          {% endif %}
        </td>
      </tr>
      <tr>
        <td>Ймовірність атаки / anomaly_score</td>
        <td>{{ prob_str }} (0-1)</td>
      </tr>
    </table>

    <div class="prob-bar">
      <div class="prob-bar-inner" style="width: {{ prob_percent }}%;"></div>
    </div>
    <div class="prob-value">
      Візуалізація рівня загрози ({{ prob_percent }}%).
    </div>

    {% if events %}
      <h2>Останні події IDS (рівень мережевих інцидентів)</h2>
      <table>
        <tr>
          <th>Час</th>
          <th>src_ip:port</th>
          <th>dest_ip:port</th>
          <th>Протокол</th>
          <th>Байти</th>
          <th>Пакети</th>
          <th>anomaly_score</th>
          <th>Статус</th>
          <th>Сигнатура</th>
          <th>Категорія</th>
          <th>Пріоритет</th>
        </tr>
        {% for e in events %}
        <tr>
          <td>{{ e.timestamp }}</td>
          <td>{{ e.src_ip }}:{% if e.src_port %}{{ e.src_port }}{% else %}-{% endif %}</td>
          <td>{{ e.dest_ip }}:{% if e.dest_port %}{{ e.dest_port }}{% else %}-{% endif %}</t
          <td>{{ e.proto }}</td>
          <td>{{ e.bytes }}</td>
          <td>{{ e.pkts }}</td>
          <td>{{ "%.4f"|format(e.anomaly_score) }}</td>
          <td>{{ e.label }}</td>
```

```
            <td>{{ e.signature }}</td>
            <td>{{ e.category }}</td>
            <td>{{ e.severity }}</td>
          </tr>
          {% endfor %}
        </table>
      {% endif %}
    {% endif %}
  </div>
</body>
</html>
"""

def get_last_stats():
    if not os.path.exists(EVE_PATH):
        return None
    with open(EVE_PATH, "r") as f:
        lines = f.readlines()
    lines = lines[-100:]
    stats_events = []
    for line in lines:
        line = line.strip()
        if not line:
            continue
        try:
            obj = json.loads(line)
        except json.JSONDecodeError:
            continue
        if obj.get("event_type") == "stats":
            stats_events.append(obj)
    if not stats_events:
        return None
    return stats_events[-1]


def get_recent_events(limit=20):
    if not os.path.exists(DB_PATH):
        return []
    conn = sqlite3.connect(DB_PATH)
    conn.row_factory = sqlite3.Row
    cur = conn.cursor()
    cur.execute(
        "SELECT timestamp, src_ip, src_port, dest_ip, dest_port, proto, "
        "pkts, bytes, anomaly_score, label, signature_id, signature, category, severity "
```

```
        "FROM events ORDER BY id DESC LIMIT ?",
        (limit,),
    )
    rows = cur.fetchall()
    conn.close()
    return rows


model = load_model(MODEL_PATH)


@app.route("/")
def index():
    last_stats = get_last_stats()
    events = get_recent_events(limit=20)

    if last_stats is None:
        return render_template_string(
            TEMPLATE,
            error="Немає подій типу 'stats' у файлі eve.json або файл недоступний.",
            uptime=None,
            pkts=None,
            bytes_total=None,
            freq=None,
            dur_minutes=None,
            prob_str=None,
            prob_percent=0,
            label=None,
            events=events,
        )

    stats = last_stats.get("stats", {})

    uptime = float(stats.get("uptime", 0.0))
    decoder = stats.get("decoder", {})
    pkts = float(decoder.get("pkts", 0.0))
    bytes_total = float(decoder.get("bytes", 0.0))

    if uptime > 0:
        freq = pkts / uptime
        bytes_per_sec = bytes_total / uptime
    else:
        freq = 0.0
        bytes_per_sec = 0.0
```

```
    dur_minutes = uptime / 60.0

    freq_feature = freq
    bytes_feature = bytes_per_sec / 100.0
    dur_feature = 1.0

    feature_vector = np.array([[freq_feature, bytes_feature, dur_feature]], dtype=float)
    prob = float(model.predict(feature_vector, verbose=0)[0][0])
    label = "ATAKA" if prob >= 0.5 else "NORMA"

    prob_percent = int(prob * 100)
    prob_str = f"{prob:.4f}"

    return render_template_string(
        TEMPLATE,
        error=None,
        uptime=f"{uptime:.1f}",
        pkts=int(pkts),
        bytes_total=int(bytes_total),
        freq=f"{freq:.2f}",
        dur_minutes=f"{dur_minutes:.2f}",
        prob_str=prob_str,
        prob_percent=prob_percent,
        label=label,
        events=events,
    )


if __name__ == "__main__":
    app.run(host="0.0.0.0", port=5000, debug=False)
```

Додаток Г
Апробація отриманих результатів

МОЛОДІЖНА НАУКОВА ЛІГА

Громадська організація «Молодіжна наукова ліга».
Номер запису в Реєстрі громадських об'єднань: 1506433.
Адреса: вул. Зодчих, буд. 40, офіс 103; м. Вінниця, Вінницька обл., 21037
Організація функціонує як відокремлений підрозділ ТОВ «UKRLOGOS Group».
ЄДРПОУ: 44574526
IBAN: UA783052990000026003016111950
Банк ВФ АТ КБ «ПриватБанк»; МФО 305299
Свідоцтво суб'єкта видавничої справи: ДК № 7172 від 21.10.2020.

ДОВІДКА

ПРО ПРИЙНЯТТЯ ТЕЗ ДО ПУБЛІКАЦІЇ

08.06.2026

Шановний(і) авторе(и):
Василічишена Валерія Сергіївна,

Організаційний комітет з радістю повідомляє, що тези «МОДУЛЬ IDS ДЛЯ ЛОКАЛЬНОЇ МЕРЕЖІ З КОМБІНОВАНИМ СИГНАТУРНИМ І ПОВЕДІНКОВИМ АНАЛІЗОМ ТРАФІКУ» прийняті до публікації в збірнику за матеріалами X Міжнародної студентської наукової конференції «Діджиталізація науки як виклик сьогодення» (12.06.2026, м. Миколаїв, Україна).

Опубліковані тези будуть доступні з 12.06.2026 за посиланням:
https://archive.liga.science/index.php/conference-proceedings/issue/view/inter-12.06.2026

. . . . . .

Електронні сертифікати учасників конференції та подяки науковим керівникам будуть доступні за посиланням вище з 12 червня. Розсилка замовлених друкованих примірників, сертифікатів та подяк відбудеться до 26 червня.

Конференцію зареєстровано Державною науковою установою «УкрІНТЕІ» в базі даних науково-технічних заходів України та інформаційному бюлетені «План проведення наукових, науково-технічних заходів в Україні» (Посвідчення № 123 від 26.01.2026).

З повагою,

Директор Молодіжної наукової ліги
Голова оргкомітету конференції
ІГОР КОРЕНЮК

МОДУЛЬ IDS ДЛЯ ЛОКАЛЬНОЇ МЕРЕЖІ З КОМБІНОВАНИМ СИГНАТУРНИМ І ПОВЕДІНКОВИМ АНАЛІЗОМ ТРАФІКУ

Василічишена Валерія Сергіївна
Бакалавр спеціальності 122 "Комп'ютерні науки"
Західноукраїнський національний університет, Україна
Науковий керівник: Осолінський Олександр Романович
ORCID ID: 0000-0002-0136-395X
к.т.н., доцент кафедри інформаційно-обчислювальних систем і управління
Західноукраїнський національний університет, Україна

Вступ

Сучасні локальні мережі містять значну кількість різнорідних пристроїв, сервісів та каналів обміну даними. В межах однієї мережі можуть одночасно працювати персональні комп'ютери, мобільні пристрої, IP-камери, мережеві принтери, сервери, засоби віддаленого доступу та IoT-пристрої. Така різнорідність ускладнює контроль безпеки, оскільки підозріла активність часто маскується під звичайні з'єднання, короткі службові запити або фоновий обмін даними. Через це для локальних мереж важливим є не тільки блокування небезпечного трафіку але постійне спостереження за поведінкою вузлів.

Одним з підходів до розв'язання цієї задачі є використання систем виявлення вторгнень IDS. Такі системи аналізують мережевий трафік, реєструють підозрілі події, формують сповіщення для адміністратора. IDS використовують сигнатурний аналіз, який порівнює пакети або мережеві події з базою відомих правил. Цей підхід добре працює для відомих атак, але може бути недостатнім у випадку модифікованих або нових загроз. Поведінковий аналіз, навпаки, дає змогу оцінювати статистичні характеристики трафіку та виявляти аномалії, але потребує правильного підбору ознак та механізмів зменшення хибних спрацювань [1], [2].

Тому для створення більш надійної системи моніторингу можна поєднати два підходи - сигнатурний для виявлення відомих загроз і поведінкової для оцінки нетипової активності.

Архітектура та програмна реалізація IDS-модуля

Запропонований модуль (рисунок 1) побудовано за модульною архітектурою куди входять окремі підсистем, які виконують захоплення трафіку, сигнатурну перевірку, виділення ознак, поведінкову класифікацію, збереження подій та відображення результатів. За апаратну основу було взято Raspberry Pi 4B [3], бо він має гігабітний мережевий інтерфейс, працює під Linux і може використовуватися як компактний IDS-сенсор в невеликій локальній мережі де є багато розумних пристроїв.

Сигнатурний аналіз реалізовано на основі Suricata. Цей компонент відповідає за аналіз мережевого трафіку за заданими правилами, виявлення відомих шаблонів атак і формування журналу подій у форматі eve.json, який є зручним для подальшої автоматизованої обробки. В системі Suricata виконує роль базового сенсора, який забезпечує первинну перевірку трафіку та передає інформацію до програмного рівня обробки [2].

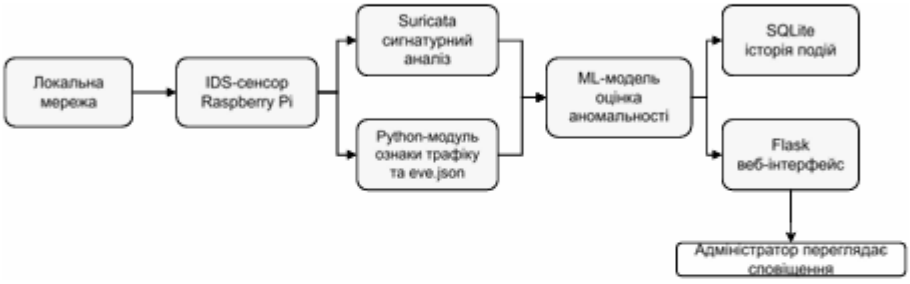


Рис. 1- **Узагальнена архітектура IDS-модуля для локальної мережі**

Алгоритм роботи системи складається з кількох послідовних етапів. Спочатку мережевий трафік захоплюється та аналізується Suricata. Далі записи з журналу eve.json зчитуються Python-модулем, фільтруються та перетворюються на набір ознак. Після цього виконується оцінювання за допомогою ML-моделі.

Результати сигнатурної перевірки й поведінкової класифікації об'єднуються в єдину подію, яка зберігається у базі даних і передається до веб-інтерфейсу.

Поведінковий аналіз реалізовано окремим Python-модулем. Його завдання полягає у зчитуванні останніх записів журналу Suricata, перетворенні технічних параметрів мережевої активності у числові ознаки та передаванні сформованого вектора на вхід моделі машинного навчання. На основі цих даних нейронна мережа обчислює ймовірність атаки, після чого система визначає стан мережі як «АТАКА» або «NORMA».

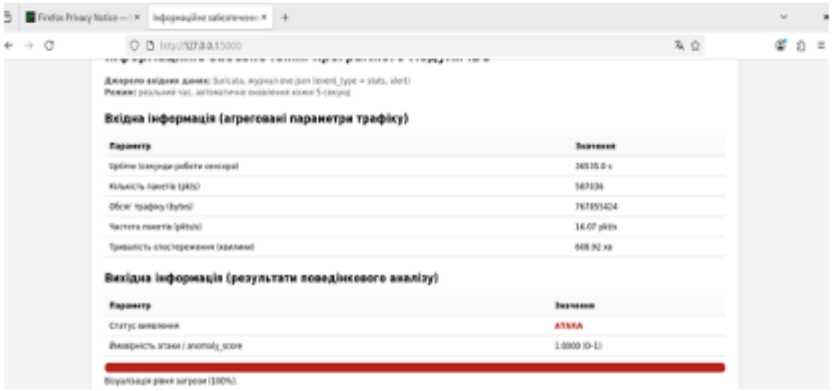


Рис. 2- **Реакція системи на підозрілу активність**

Для реалізації моделі використано бібліотеки TensorFlow і Keras. Нейронна мережа доповнює сигнатурний аналіз, якщо відома атака описана в базі правил, її може виявити Suricata. Якщо ж пакет не містить очевидної сигнатури, але поведінка вузла відрізняється від типової, підвищену оцінку ризику формує поведінковий модуль [4].

Висновки. Сформовано та реалізовано IDS-модуль для локальної мережі, який поєднує сигнатурний і поведінковий аналіз мережевого трафіку. Сигнатурна частина на базі Suricata забезпечує виявлення відомих загроз та формування структурованих журналів подій, а поведінковий модуль на Python з

використанням нейронної мережі оцінює статистичні відхилення у трафіку. Для зберігання інцидентів використано SQLite, а для перегляду результатів розроблено веб-інтерфейс на Flask. Запропонований підхід може бути використаний для створення навчальних і практичних стендів-сенсорів з моніторингу безпеки локальних мереж.

Список використаних джерел:

1. Guide to Intrusion Detection and Prevention Systems (IDPS). URL: https://nvlpubs.nist.gov/nistpubs/legacy/sp/nistspecialpublication800-94.pdf.
2. What is Suricata. Suricata User Guide. URL: https://docs.suricata.io/en/latest/what-is-suricata.html.
3. Raspberry Pi 4 Model B Datasheet. Raspberry Pi Ltd., 2024. URL: https://pip.raspberrypi.com/documents/RP-008341-DS-raspberry-pi-4-datasheet.pdf.
4. Kwon H.-Y., Kim T., Lee M.-K. Advanced Intrusion Detection Combining Signature-Based and Behavior-Based Detection Methods // Electronics. 2022. Vol. 11, no. 6. Art. 867. DOI: https://doi.org/10.3390/electronics11060867.