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

ДОПУСТИТИ ДО ЗАХИСТУ
Завідувач кафедри кібербезпеки

_____Анна ІЛЬЄНКО
"_____"_____ 20__ р.

КВАЛІФІКАЦІЙНА РОБОТА

(ПОЯСНЮВАЛЬНА ЗАПИСКА)

ЗДОБУВАЧА ОСВІТНЬОГО СТУПЕНЯ "МАГІСТР"

Тема: Програмний модуль виявлення кібератак в комп'ютерних системах

Виконавець: Артем ТРІЩУН

Керівник: к.т.н., доцент Анна ІЛЬЄНКО

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

Київ 2024

ДЕРЖАВНЕ НЕКОМЕРЦІЙНЕ ПІДПРИЄМСТВО
«ДЕРЖАВНИЙ УНІВЕРСИТЕТ
«КИЇВСЬКИЙ АВІАЦІЙНИЙ ІНСТИТУТ»

Факультет комп'ютерних наук та технологій
Кафедра кібербезпеки
Освітній ступінь магістр
Спеціальність 125 «Кібербезпека та захист інформації»
Освітньо-професійна програма «Безпека інформаційних і комунікаційних систем»

ЗАТВЕРДЖУЮ
Завідувач кафедри кібербезпеки

_____Анна ІЛЬЄНКО
«30»___08_____2024 р.

ЗАВДАННЯ

на виконання кваліфікаційної роботи

Тріщуна Артема Володимировича

1. Тема кваліфікаційної роботи: Програмний модуль виявлення кібератак в комп'ютерних системах затверджена наказом ректора від 30.08.2024 р. №1695/ст.
2. Термін виконання роботи: з 30.08.2024 по 15.12.2024
3. Вихідні дані: провести дослідження сучасних загроз та типів кібератак; провести аналіз методів та програмних засобів для виявлення кібератак; на основі дослідження та аналізу розробити архітектуру програмного модуля для виявлення кіберзагроз; провести тестування та оцінки ефективності розробленого програмного модуля для виявлення кібератак у комп'ютерних системах.
4. Зміст пояснювальної записки: Огляд сучасних методів виявлення кібератак, Теоретичний опис та архітектура програмного модуля, Програмна реалізація модуля.ів
5. Перелік обов'язкового графічного (ілюстративного) матеріалу: презентація, діаграми.

6. Календарний план-графік

№ з/п	Завдання	Термін виконання	Підпис керівника
1.	Проведення аналізу сучасних загроз і методів виявлення кібератак	30.08.2024 – 12.09.2024	*Виконано*
2.	Вибір і обґрунтування методів для програмного модуля	12.09.2024 – 17.09.2024	*Виконано*
3.	Розробка архітектури та структурної схеми програмного модуля	18.09.2024 – 30.09.2024	*Виконано*
4.	Програмна реалізація модулів збору, обробки та аналізу даних	1.10.2024 – 30.10.2024	*Виконано*
5.	Тестування розробленого модуля і аналіз результатів	1.11.2024 – 22.11.2024	*Виконано*

7. Дата видачі завдання: «30» __08__ 2024 р.

Керівник кваліфікаційної роботи: _____ Анна ІЛЬЄНКО

(підпис керівника) (П.І.Б.)

Завдання прийняв до виконання: _____ Артем ТРІЩУН

(підпис здобувача вищої освіти) (П.І.Б.)

РЕФЕРАТ

Пояснювальна записка до кваліфікаційної роботи: «Програмний модуль системи виявлення кібератак» складається зі вступу, основної частини, що містить 3 розділи, загального висновоку, додатків та списку використаних джерел. Загальний обсяг роботи – 137 сторінок. Робота містить 26 рисунків та 13 таблиць. Список використаних джерел включає 32 джерел.

Об'єкт дослідження: процеси виявлення, моніторингу та аналізу кіберзагроз у комп'ютерних системах з використанням спеціалізованих програмних модулів і алгоритмів.

Предмет дослідження: комп'ютерні системи та мережі, їхні інформаційні ресурси, що піддаються загрозам кібератак, несанкціонованого доступу та порушенням безпеки.

Мета роботи: розробка програмного модуля для виявлення та аналізу кіберзагроз у комп'ютерних системах з використанням алгоритмів машинного навчання та моніторингових засобів.

Методи: методи машинного навчання та аналізу даних для виявлення кіберзагроз, методи моніторингу та оцінки ефективності виявлення загроз, а також методи обробки та класифікації даних.

Практична цінність: практична цінність роботи полягає в розробці програмного модуля для виявлення кібератак у комп’ютерних системах мовою програмування Python, що забезпечує підвищений рівень інформаційної безпеки. Модуль здатний виявляти атаки (зокрема DDoS, SQL-ін'єкції, XSS, CSRF та аномалії у мережевому трафіку) у режимі реального часу, що дозволяє своєчасно реагувати на загрози та мінімізувати ризики несанкціонованого доступу.

НАУКОВА НОВИЗНА - наукова новизна роботи полягає у розробці авторського програмного модуля для виявлення кібератак, що об’єднує сигнатурний метод із методами машинного навчання. Такий комбінований підхід

забезпечує у 1.5-2 рази кращу точність виявлення загроз порівняно з аналогами, завдяки:

— Інтеграції сигнатурного підходу з машинним навчанням, що дозволяє точніше ідентифікувати навіть нові загрози;

— Використанню кількох моделей машинного навчання, які підвищують ефективність роботи системи з різними типами атак;

— Обробці даних у реальному часі для миттєвої реакції на кіберзагрози;

— Оптимізації метрик виявлення, що дозволяє зменшити кількість хибних спрацьовувань і досягти високої точності.

Ключові слова: ПРОГРАМНИЙ МОДУЛЬ, КІБЕРБЕЗПЕКА, МАШИННЕ НАВЧАННЯ, КІБЕРАТАКА, ШТУЧНИЙ ІНТЕЛЕКТ.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

AUC (Area Under Curve)	–	площа під ROC-кривою, міра якості класифікації.
CSRF (Cross-Site Request Forgery)	–	міжсайтове підроблення запитів для виконання несанкціонованих дій.
DDoS (Distributed Denial of Service)	–	розподілена атака відмови в обслуговуванні.
DoS (Denial of Service)	–	відмова в обслуговуванні.
FPR (False Positive Rate)	–	частка хибних виявлень.
IDS (Intrusion Detection System)	–	система виявлення вторгнень.
IPS (Intrusion Prevention System)	–	система запобігання вторгненням.
ML (Machine Learning)	–	машинне навчання.
MLP (Multilayer Perceptron)	–	багатошаровий персептрон, тип нейронної мережі.
ROC Curve (Receiver Operating Characteristic)	–	крива робочих характеристик приймача для оцінки ефективності моделі.
SQL Injection	–	ін'єкція SQL-коду для несанкціонованого доступу до бази даних.
SVM (Support Vector Machine)	–	метод опорних векторів, алгоритм машинного навчання.
TPR (True Positive Rate)	–	частка правильних виявлень

XSS (Cross-Site Scripting)	– міжсайтовий скриптинг для виконання шкідливого коду на стороні клієнта.

ВСТУП

Сучасний розвиток інформаційних технологій, а також постійне зростання кількості кіберзагроз роблять питання безпеки комп'ютерних систем надзвичайно актуальним у різних галузях, таких як фінансовий сектор, охорона здоров’я, промисловість, телекомунікації, державне управління та інших. Різноманітні типи атак, такі як DDoS, XSS, SQL-ін'єкції, CSRF, фішинг та програми-вимагачі, стали серйозними викликами для безпеки інформаційних ресурсів, оскільки вони здатні обходити традиційні засоби захисту, завдаючи значних збитків у випадку їх реалізації.

Поява нових вразливостей у програмному забезпеченні, розширення обсягів даних, а також активне впровадження IoT пристроїв створюють ще більше ризиків для компаній та організацій, оскільки вплив атаки може поширюватися на всі взаємопов’язані системи. На додаток до цього, хмарні обчислення та віртуальні середовища також збільшують кількість потенційних вразливостей.

Загроза кібербезпеці ускладнюється не тільки через збільшення обсягів даних, але й завдяки зростанню технологій автоматизації, які можуть використовувати зловмисники для одночасного впливу на багато систем. Кожна секунда затримки у виявленні загрози може призвести до серйозних наслідків: втрати важливої інформації, зупинки бізнес-процесів, компрометації конфіденційних даних та великих фінансових втрат. Це вимагає сучасних, ефективних і швидкодійних рішень для моніторингу та захисту мереж у реальному часі, які здатні ідентифікувати загрози навіть у складних випадках.

Відтак, необхідність розробки інтегрованих програмних рішень, що поєднують машинне навчання з традиційними методами аналізу загроз, стає ще більш актуальною. Застосування таких рішень дозволяє не тільки підвищити ефективність кіберзахисту, але й досягти більшої точності у

виявленні атак, забезпечити проактивний моніторинг та зменшити час реагування на потенційні загрози.

Запропонований програмний модуль, який об'єднує сигнатурний метод із методами машинного навчання, забезпечує підвищену ефективність виявлення кіберзагроз завдяки швидкому аналізу та адаптації до нових типів атак. Це дозволяє підвищити точність і зменшити час обробки атак, що особливо важливо для захисту інформаційних ресурсів у динамічному середовищі сучасних мереж.

Дослідження і розробка в цій галузі сприяє не тільки підвищенню загального рівня кібербезпеки, а й розширенню наукових знань і практичних навичок у сфері аналізу шкідливого програмного забезпечення, що, безумовно, є актуальною і значущою проблемою сучасності.

Мета роботи - розробка програмного модуля для виявлення та аналізу кіберзагроз у комп'ютерних системах з використанням алгоритмів машинного навчання та моніторингових засобів.

Виходячи з поставленої мети завданнями кваліфікаційної роботи є:

— Дослідження сучасних загроз та типів кібератак;

— Аналіз методів та програмних засобів для виявлення кібератак;

— Розробка архітектури програмного модуля для виявлення кіберзагроз;

— Проведення тестування та оцінки ефективності розробленого програмного модуля для виявлення кібератак у комп'ютерних системах.

Об'єкт дослідження – процеси виявлення, моніторингу та аналізу кіберзагроз у комп'ютерних системах з використанням спеціалізованих програмних модулів і алгоритмів.

Предмет дослідження – комп'ютерні системи та мережі, їхні інформаційні ресурси, що піддаються загрозам кібератак, несанкціонованого доступу та порушенням безпеки.

Методи – методи машинного навчання та аналізу даних для виявлення кіберзагроз, методи моніторингу та оцінки ефективності виявлення загроз, а

також методи обробки та класифікації даних.

НАУКОВА НОВИЗНА – полягає у створенні програмного модуля виявлення кібератак, який використовує комбінований підхід, що поєднує сигнатурний метод та методи машинного навчання для ефективнішого виявлення загроз. Це дозволяє досягти покращених показників порівняно з аналогами, зокрема підвищення точності виявлення та зменшення часу обробки атак за рахунок:

- Комбінованого підходу до виявлення загроз
- Застосування кількох моделей машинного навчання
- Реалізації обробки даних у реальному часі
- Використання метрик і оптимізації

ПРАКТИЧНА ЦІННІСТЬ – практична цінність роботи полягає в розробці програмного модуля для виявлення кібератак у комп'ютерних системах мовою програмування Python, що забезпечує підвищений рівень інформаційної безпеки. Модуль здатний виявляти атаки у режимі реального часу, що дозволяє своєчасно реагувати на загрози та мінімізувати ризики несанкціонованого доступу.

Апробація.

1. Тріщун А.В. Програмний модуль виявлення кібератак в комп'ютерних системах /А.В.Ільєнко, А.В.Тріщун // Perspectives of contemporary science: theory and practice: IX International Scientific and Practical Conference, 14-16 October 2024: тези доповіді. – Львів (Україна), 2024. – P. 406-410

РОЗДІЛ 1
ОГЛЯД СУЧАСНИХ МЕТОДІВ ВИЯВЛЕННЯ КІБЕРАТАК

1.1 Класифікація кібератак

Кібератака – це будь-яка навмисна спроба викрасти, викрити, змінити, вивести з ладу або знищити дані, програми чи інші активи шляхом несанкціонованого доступу до мережі, комп’ютерної системи чи цифрового пристрою.

1.1.1 Типи кібератак

Кібератаки бувають різними за своєю природою, метою та методами здійснення. Розкриття конфіденційної інформації, спорожнення банківських рахунків та крадіжка особистих даних – це лише деякі її жахливі наслідки.

Кібератаки можна поділити на:

1) Атаки на основі програмного забезпечення:

Вірус – це невелика програмна програма, яка поширюється з одного комп'ютера на інший і перешкоджає роботі комп'ютера. Комп'ютерний вірус може пошкодити або видалити дані на комп'ютері, поширити вірус на інші комп'ютери за допомогою програми електронної пошти або навіть видалити всі дані на жорсткому диску.

Хробак/Комп’ютерний черв’як – це різновид шкідливого програмного забезпечення, яке спрямоване на знищення певних програм або навіть усієї системи шляхом пошкодження файлів. Крім того, він відтворює себе заздалегідь визначеним чином і часто націлюється на інші комп’ютери в мережі, якщо вони присутні [1].

Троянські програми – це програми, яка надає стороннім доступ до комп'ютера для здійснення будь-яких дій на місці призначення без попередження самого власника комп'ютера або висилає за певною адресою

зібрану інформацію. При цьому вона, як правило, дається взнаки за що-небудь мирне і надзвичайно корисне [2].

Шкідливе ПЗ – програмне забезпечення, яке перешкоджає роботі комп'ютера, збирає конфіденційну інформацію або отримує доступ до приватних комп'ютерних систем. Може проявлятися у вигляді коду, скрипту, активного контенту, і іншого програмного забезпечення.

2) Атаки на основі мережі:

DoS/DDoS – атаки типу «відмова в обслуговуванні» блокують або порушують здатність організації чи підприємства використовувати власні ресурси, такі як пропускна здатність мережі, системні ресурси і ресурси додатків. Розподілена атака на відмову в обслуговуванні – це більш просунута форма DoS-атаки, коли цільова мережа переповнюється запитами не від одного сервера чи машини, а від кількох точок атаки.

Man-in-the-Middle – це форма кібератаки, під час якої для перехоплення даних використовують методи, що дають змогу впровадитися в наявне підключення або процес зв'язку. Зловмисник може бути пасивним слухачем у вашій розмові, який непомітно краде якісь відомості, або активним учасником, змінюючи зміст ваших повідомлень або видаючи себе за людину або систему, з якими ви, на вашу думку, розмовляєте [3].

Sniffing – дана атака полягає в тому, що зловмисник проникає в мережевий потік даних і читає, відстежує або перехоплює повні пакети даних, що передаються між клієнтом і сервером. Хакер, який перехоплює мережевий пакет, що містить незашифровану інформацію, може завдати серйозної шкоди організації чи організації, яка володіє даними. Зламані дані можуть включати конфіденційну інформацію, як-от облікові дані облікового запису, банківські реквізити та різні види особистої інформації.

3) Атаки на основі соціальної інженерії:

Фішинг – це форма атаки з використанням соціальної інженерії, в ході якої зловмисник, маскуючись під надійний суб'єкт, виманює конфіденційну інформацію жертв.

Претекстинг – це використання сфабрикованої історії або приводу, щоб завоювати довіру жертви та обманом або змусити її поділитися конфіденційною інформацією, завантажити зловмисне програмне забезпечення, надіслати гроші злочинцям або іншим чином завдати шкоди собі чи організації, на яку вони працюють [4].

Baiting Attack (Приманка) – стратегія, яка використовується в соціальній інженерії, коли людину спокушають оманливою обіцянкою, яка викликає її цікавість або жадібність. Приманка – це коли зловмисник залишає USB-накопичувач із шкідливим корисним навантаженням у вестибюлях або на парковках у надії, що хтось із цікавості вставить його в пристрій, у цей час може бути розгорнуто шкідливе програмне забезпечення, яке воно містить [5].

4) Атаки на веб-додатки:

SQL Injection – це тип ін’єкційної атаки, яка дає змогу виконувати зловмисні оператори SQL. Ці оператори керують сервером бази даних за веб-програмою. Зловмисники можуть використовувати вразливості SQL Injection, щоб обійти заходи безпеки програми. Вони можуть обійти автентифікацію та авторизацію веб-сторінки або веб-додатку та отримати вміст усієї бази даних SQL. Вони також можуть використовувати SQL Injection для додавання, зміни та видалення записів у базі даних [6, 7].

Міжсайтовий скриптинг/Cross-Site Scripting – це тип ін’єкцій, під час яких шкідливі сценарії впроваджуються на безпечні та надійні веб-сайти. XSS-атаки відбуваються, коли зловмисник використовує веб-програму для надсилання шкідливого коду, як правило, у формі сценарію на стороні браузера, іншому кінцевому користувачеві [8, 9].

Міжсайтова підробка запитів/Cross-Site Request Forgery – це атака, яка змушує автентифікованих користувачів надіслати запит до веб-додатку, у якому вони наразі автентифіковані. Атака CSRF використовує вразливість у веб-програмі, якщо вона не може відрізнити запит, створений окремим користувачем, від запиту, створеного користувачем без його згоди.

5) Атаки на рівні операційної системи:

Privilege Escalation – Підвищення привілеїв відбувається, коли зловмисник отримує розширений доступ і права адміністратора в системі, використовуючи вразливості в її безпеці. Маніпулюючи дозволами ідентифікації для надання собі додаткових прав, зловмисники можуть здійснювати шкідливі дії, що може призвести до серйозних наслідків.

Руткіт – це програма або набір шкідливих програмних засобів, які надають зловмисникам віддалений доступ до комп'ютера або іншої системи та контроль над ними. Руткіти відкривають бекдор у системах жертв для впровадження шкідливого програмного забезпечення, зокрема вірусів, програм-вимагачів, програм-клавіатурних шпигунів чи інших типів зловмисного програмного забезпечення , або використовувати систему для подальших атак безпеки мережі. Руткіти часто намагаються запобігти виявленню зловмисного програмного забезпечення, дезактивуючи кінцеві точки захисту від зловмисного програмного забезпечення та антивірусного програмного забезпечення [10].

Програми-бекдори – це програми, які дозволяють кіберзлочинцям або зловмисникам отримувати віддалений доступ до комп'ютерів. Бекдори можуть бути встановлені як в програмних, так і в апаратних компонентах. Багато бекдор-програм використовують магістраль IRC, отримуючи команди від звичайних клієнтів чату IRC[11-13].

1.1.2 Приклади та наслідки реальних кібератак

Розуміння різних типів атак допомагає спеціалістам з кібербезпеки передбачати можливі загрози, вчасно їх виявляти та ефективно з ними боротися. Класифікація кібератак дозволяє створювати більш точні та ефективні системи захисту, які можуть адаптуватися до нових загроз і методів атак. Також розробив таблицю в якій описав приклади кібератак та їх наслідки[14].

Таблиця 1.1

Опис прикладів та наслідків реальних кібератак

Наслідки	Опис	Приклад
1	2	3
Економічні збитки	Кібератаки можуть завдати значних економічних втрат організаціям і приватним особам.	У 2017 році відбулася одна з найбільших атак програм-вимагачів. 12 травня WannaCry заблокувала десятки тисяч комп'ютерів по всьому світу, в тому числі в державних установах і великих компаніях.
Компрометація даних	Кібератаки можуть призвести до витоку конфіденційних даних, таких як фінансова інформація, особисті дані користувачів, комерційна таємниця.	Атака хакерів на готелі Marriott і Starwood Hotels Group залишалася непоміченою роками. Лише в 2018 році фахівцям вдалося виявити зловмисників в мережі. До цього моменту дані приблизно 339 млн гостей були скомпрометовані.
Порушення операційної діяльності	Атаки можуть порушити роботу важливих систем, що може призвести до зупинки виробничих процесів, переривання обслуговування клієнтів та інших серйозних наслідків.	Одна з найперших і найбільших кіберзагроз – вірус Melissa. У 1999 році програміст Девід Лі Сміт надсилав вірусний файл для відкриття через Microsoft Word. Після чого вірус активувався, завдаючи серйозної шкоди сотням компаній, у тому числі Microsoft. За оцінками, ремонт уражених систем коштував компанії 80 млн доларів.

Закінчення таблиці 1.1

1	2	3
Загроза національній безпеці	Деякі кібератаки можуть мати серйозні наслідки для національної безпеки, особливо якщо вони спрямовані на критичну інфраструктуру (енергетика, транспорт, зв'язок).	Атака на електромережі України у 2015 році стала першою кібератакою на об’єкти енергетики. Ця актака дозволила зловмисникам вивести з ладу основні елементи IT-інфраструктури.
Репутаційні втрати	Компанії, які стали жертвами кібератак, можуть втратити довіру клієнтів і партнерів, що негативно вплине на їх репутацію і ринкову позицію.	У 2013 році компанія Adobe стикнулася з атакою хакерів, яка призвела до масштабного витоку даних. Спочатку вважалося, що кібератака зламала дані 2,9 млн користувачів. Проте пізніше компанія заявила, що витік даних зачепив понад 38 млн.

1.2 Методи та способи виявлення кібератак

Для забезпечення безпеки комп’ютерних систем і захисту даних необхідно не лише впроваджувати захисні механізми, але й мати здатність своєчасно виявляти та реагувати на загрози. З розвитком інформаційних технологій зростають не лише можливості, але й ризики. Кібератаки можуть призвести до серйозних наслідків, таких як фінансові втрати, витік конфіденційної інформації та порушення нормальної діяльності організацій. Тому ефективне виявлення кібератак стає надзвичайно важливим для забезпечення кібербезпеки. Виявлення атак на ранніх стадіях дозволяє мінімізувати їхні наслідки та швидко відновити нормальну роботу систем[15].

Методи виявлення кібератак можуть варіюватися від простих сигнатурних методів до складних систем на основі машинного навчання та штучного інтелекту.

1.2.1 Мета виявлення кібератак

Виявлення кібератак є однією з основних складових кібербезпеки, що дозволяє своєчасно реагувати на загрози та мінімізувати їхні негативні наслідки. Для наочного розуміння створив таблицю в якій вказав основні цілі виявлення кібератак.

Таблиця 1.2

Опис цілей виявлення кібератак

Ціль виявлення кібератаки	Опис	Приклад
1	2	3
Захист інформаційних активів	Основна мета виявлення кібератак полягає в захисті конфіденційних і важливих даних від несанкціонованого доступу та викрадення.	Вчасне виявлення атаки на базу даних допомагає запобігти викраденню фінансової інформації користувачів.
Зниження шкоди від атак	Виявлення кібератак на ранніх стадіях дозволяє мінімізувати шкоду, яку можуть завдати зловмисники, зменшивши час їх перебування в системі.	Виявлення шкідливого програмного забезпечення на початковому етапі його дії може запобігти поширенню вірусу по всій мережі.
Забезпечення безперервності бізнесу	Виявлення кібератак допомагає підтримувати безперебійну роботу бізнес-процесів, мінімізуючи простої та інші збої, спричинені атаками.	Вчасне виявлення DDoS атаки дозволяє швидко реагувати та підтримувати доступність веб-сайтів та онлайн-сервісів.
Зменшення фінансових втрат	Ефективне виявлення кібератак допомагає зменшити фінансові втрати, пов'язані з витоком даних, відновленням систем, штрафами та втратами репутації.	Виявлення фішингової атаки на банківську систему допомагає запобігти крадіжці коштів з рахунків клієнтів.

Закінчення таблиці 1.2

1	2	3
Підвищення довіри користувачів та партнерів	Постійний моніторинг та виявлення кібератак сприяє підвищенню довіри з боку клієнтів, партнерів та інших зацікавлених сторін до безпеки компанії.	Впровадження системи виявлення атак та прозорість у повідомленнях про заходи безпеки зміцнює репутацію компанії як надійного партнера.
Відповідність нормативним вимогам	Виявлення та своєчасне реагування на кібератаки є обов'язковим елементом для відповідності багатьом стандартам та нормативним вимогам з кібербезпеки.	Дотримання вимог GDPR, що стосуються захисту персональних даних, вимагає впровадження систем виявлення та реагування на кібератаки.
Покращення механізмів захисту	Аналіз виявлених атак дозволяє вдосконалювати існуючі механізми захисту та розробляти нові стратегії для протидії майбутнім загрозам.	Виявлення нових типів шкідливого ПЗ допомагає розробникам антивірусних програм покращити свої продукти.

1.2.2 Методи виявлення кібератак

Для ефективного виявлення кібератак використовуються різні методи та підходи, кожен з яких має свої особливості, переваги та недоліки. Розуміння цих методів є ключовим для побудови надійних систем кібербезпеки, здатних своєчасно виявляти та реагувати на загрози.

Методи виявлення кібератак бувають наступними:

1. Сигнатурний – даний метод дозволяє описати кібератаку за допомогою набору правил, або за допомогою формальної моделі, в якості якої може виступати символьний рядок, семантичний вираз спеціальною мовою, тощо. Суть цього методу полягає у використанні спеціалізованої

бази даних шаблонів кібератак для пошуку дій, які підпадають під визначення «кібератака».

Сигнатурний метод може захистити від вірусної або хакерської кібератаки, коли її сигнатура вже відома і внесена до бази даних. Якщо ж мережа зазнає першої атаки ззовні, перше зараження ще невідоме, і в базі даних просто не вистачає сигнатури для його пошуку, тому система не зможе сигналізувати про небезпеку, оскільки вважає атакуючу активність легітимною.

Сигнатурний метод також в свою чергу поділяється на інші. Одним із найпоширеніших сигнатурних методів виявлення атак є метод контекстного пошуку певної множини символів у вихідних даних. Цей метод дозволяє ефективно виявляти атаки через аналіз мережевого трафіку, оскільки він забезпечує можливість точно визначити параметри сигнатури, яку необхідно виявити у потоці даних.

Ще одним методом є метод аналізу станів, який формує сигнатури атак у вигляді послідовності переходів інформаційної системи з одного стану в інший. Кожен такий перехід пов'язаний із настанням певних подій в ІС, що визначаються параметрами сигнатури атаки.

Також є сигнатурний метод, що базується на експертних системах, він дозволяють описувати моделі атак природною мовою з високим рівнем абстракції. Експертна система складається з бази фактів та бази правил. Факти являють собою вихідні дані про роботу ІС, а правила – методи логічного висновку про атаку на основі наявної бази фактів.

2. Метод виявлення на основі аномалій – даний метод полягає у знаходженні та ідентифікації елементів, подій або спостережень, які відхиляються від очікуваної поведінки або інших елементів набору даних.

У контексті виявлення зловживань або вторгнень у мережу, основний інтерес представляють несподівані сплески активності, а не просто рідкісні події. Ці сплески не відповідають загальному статистичному визначенню викидів як рідкісних об'єктів, тому багато

методів виявлення викидів, особливо неконтрольовані алгоритми, неефективні без відповідного збору даних.

Методів виявлення аномалій поділяються на 3 основні типи:

1. Неконтрольовані методи – дані методи визначають аномалії на непозначеному наборі даних, виходячи з припущення, що більшість зразків у наборі є нормальними. Вони шукають зразки, які найменше відповідають решті набору даних.
2. Контрольовані методи – дані методи потребують позначеного набору даних, де зразки класифіковані як «нормальні» або «аномальні». Вони навчають класифікатор на цьому наборі даних, хоча ключовою проблемою є незбалансований характер виявлення аномалій.
3. Напівконтрольовані методи – дані методи будують модель нормальної поведінки на основі позначеного навчального набору даних, а потім перевіряють, наскільки ймовірно, що тестовий екземпляр було створено цією моделлю.

3. Метод машинного навчання та ШІ – даний метод є дуже результативним в наш час оскільки ШІ здатен аналізувати величезні обсяги даних із різноманітних джерел, виявляючи закономірності активності всередині організації. Він може відстежувати час і місце входу в систему, обсяг мережевого трафіку, а також використовувані пристрої та хмарні програми. Зрозумівши, які дії є типовими для співробітників, ШІ здатен виявити аномалії, які можуть свідчити про потенційні загрози і потребують додаткового дослідження[16].

При цьому для забезпечення конфіденційності дані однієї організації не використовуються для навчання ШІ в інших компаніях. Замість цього, ШІ використовує синтезовану глобальну інформацію про загрози, зібрану з багатьох організацій, для підвищення ефективності аналізу.

Використовуючи алгоритми машинного навчання, ШІ постійно навчається на основі даних, які оцінює система. Коли генеративний ШІ виявляє відомі кіберзагрози, такі як зловмисне програмне забезпечення, він може надати контекст для аналізу загроз, роблячи його зрозумілішим. Наприклад, ШІ може створювати новий текст або зображення, щоб детально описати, що відбувається під час атаки.

4. Комбінований – даний метод виявлення кібератак поєднує різні підходи для підвищення ефективності виявлення та зниження кількості хибних позитивних спрацьовувань.

Цей метод використовує переваги декількох підходів, таких як сигнатурний аналіз, аналіз аномалій та машинне навчання, для створення більш надійної та точної системи виявлення загроз.

З сигнатурного аналізу цей метод використовує бази даних відомих шаблонів або сигнатур атак для порівняння поточної активності з відомими атаками.

За допомогою аналізу аномалій даний метод виявляє відхилення від нормальної поведінки системи або мережі шляхом створення моделі нормальної активності і відстеження будь-яких аномалій. Це дозволяє виявляти нові, раніше невідомі атаки.

А завдяки машинному навчанню та ШІ даний метод автоматично виявляє аномалії та атаки, аналізуючи великі обсяги даних і виявляючи складні шаблони.

Після дослідження методів виявлення кібератак можна зробити висновки по кожному з методів:

Таблиця 1.3

Висновки по методам виявлення кібератак

Назва методу	Принцип роботи	Переваги	Недоліки	Приклади інструментів
1	2	3	4	5
Сигнатурний	Використання попередньо відомих шаблонів або сигнатур атак для порівняння з базою даних відомих сигнатур.	Висока точність для відомих атак, низький рівень хибно позитивних спрацьовувань.	Невразливість до нових, невідомих атак (zero-day атак), потреба в постійному оновленні бази сигнатур.	Антивірусне ПЗ, системи виявлення вторгнень (IDS), системи запобігання вторгнень (IPS).
На основі аномалій	Виявлення відхилень від нормальної поведінки системи або мережі шляхом створення моделі нормальної активності і відстеження будь-яких аномалій.	Можливість виявлення нових і невідомих атак, адаптивність до змін у поведінці системи.	Високий рівень хибнопозитивних спрацьовувань, складність налаштування та підтримки.	Системи мережевого моніторингу, інструменти аналізу трафіку.
Машинного навчання та ШІ	Використання алгоритмів машинного навчання та штучного інтелекту для автоматичного виявлення аномалій та атак, аналізуючи великі обсяги даних і виявляючи складні шаблони.	Висока ефективність в умовах великих даних, здатність до самонавчання та адаптації до нових загроз.	Необхідність у великій кількості даних для навчання, складність інтерпретації результатів, висока вартість впровадження та підтримки.	Системи на основі deep learning, автоматизовані платформи кібербезпеки.

Закінчення таблиці 1.3

1	2	3	4	5
Комбінований	Поєднання різних підходів для підвищення ефективності виявлення атак, об’єднуючи сигнатурний аналіз, аналіз аномалій та машинне навчання.	Більш висока точність, зниження кількості хибнопозитивних спрацьовувань, можливість виявлення широкого спектра атак.	Вища складність впровадження і обслуговування, потреба в координації різних методів та інструментів.	Комплексні системи захисту, що включають різні методи виявлення атак, інтегровані платформи кібербезпеки.

1.2.3 Порівняння методів виявлення кібератак

Для порівння методів виявлення кібератак потрібно чітко прохарактеризувати такі важливі аспекти, як точність виявлення, вразливість до нових атак, простота впровадження та налаштування, вимоги до ресурсів, адаптивність та вартість впровадження та підтримки.

Точність виявлення є одним з ключових критеріїв, що визначає ефективність методу. Вона показує, наскільки метод може точно ідентифікувати реальні загрози, мінімізуючи кількість хибнопозитивних спрацьовувань.

Вразливість до нових атак оцінює здатність методу виявляти атаки, про які раніше не було відомо. Це особливо важливо в умовах постійно зростаючої кількості нових загроз.

Простота впровадження та налаштування враховує складність інтеграції методу в існуючу інфраструктуру організації та ресурси, необхідні для його налаштування.

Вимоги до ресурсів визначають обчислювальні потужності, пам’ять та обсяг зберігання даних, необхідні для ефективного функціонування методу.

Адаптивність показує, наскільки метод здатний адаптуватися до змін у поведінці системи або мережі.

І врешті решт, останнім, але не менш важливим аспектом є вартість впровадження та підтримки методу. Вона включає як фінансові витрати на початкове впровадження, так і постійні витрати на підтримку та оновлення.

Враховуючи всі ці фактори, можна провести детальний аналіз та порівняння методів виявлення кібератак для визначення найбільш ефективних підхідів для забезпечення кібербезпеки в різних умовах.

Для простоти розуміння розробив таблицю з описаними вище характеристиками.

Таблиця 1.4

Опис характеристик методів виявлення кібератак

Метод	Точність виявлення	Вразливість до нових атак	Простота впровадження та налаштування	Вимоги до ресурсів	Адаптивність	Вартість впровадження та підтримки
Сигнатурний	Висока для відомих атак	Низька	Відносно проста	Низькі	Низька	Низька
На основі аномалій	Висока для нових атак	Висока	Складна	Високі	Висока	Висока
Машинного навчання та ШІ	Дуже висока при достатній кількості даних	Висока	Дуже складна	Дуже високі	Дуже висока	Дуже висока
Комбінований	Висока	Висока	Дуже складна	Високі	Висока	Висока

1.3 Сучасні програмні рішення виявлення кібератак

Захист від кіберзагроз вимагає постійного вдосконалення та використання передових технологій. Програми для виявлення кібератак застосовують сучасні техніки та алгоритми для швидкого виявлення аномалій у мережі.

Сучасні системи здатні навчатися розпізнавати незвичну активність та відхилення від стандартної поведінки системи. Вони використовують методи машинного навчання для аналізу великих обсягів даних і набувають здатності ідентифікувати нові види загроз.

Однією з основних функцій таких програм є детальний аналіз кіберзагроз. Вони застосовують технології штучного інтелекту та обробки великих обсягів даних для виявлення прихованих зв'язків і шаблонів у кібератаках. Використання алгоритмів машинного навчання дозволяє швидше виявляти та реагувати на нові види загроз, що особливо важливо в умовах зростання кількості та складності кібератак.

Ще однією важливою особливістю сучасних програм для виявлення кібератак є їхня адаптивність та постійне оновлення. Оскільки кіберзагрози постійно змінюються та вдосконалюються, ці програми повинні регулярно оновлюватися, додаючи нові сигнатури та алгоритми для ефективного виявлення останніх видів загроз.

Взагалі, у нас час достатньо різних програмних рішень тому буде краще їх класифікувати, а потім описувати, до таких відносяться:

1. Комерційні програмні рішення (програмне забезпечення, створене комерційною організацією з метою отримання прибутку від його використання іншими):

    1.1 Symantec Endpoint Protection – це комплекс програмного забезпечення для безпеки, що включає функції запобігання вторгненням, брандмауер та захист від шкідливих програм.

    Дане програмне забезпечення має багато функцій до яких входять:

    — Сканування комп’ютера на наявність вірусів, шпигунського та

іншого шкідливого програмного забезпечення, забезпечуючи їх виявлення та видалення.

— Блокування відомих загроз та захист мережі від атак за допомогою системі запобіганню вторгнень.

— Можливість створення та застосування політик брандмауера для контролю мережевого трафіку та захисту від несанкціонованого доступу.

— Можливість становлення політики щодо використання зовнішніх пристроїв, таких як USB-накопичувачі, щоб запобігти витоку даних.

— Захист системи від атак, що використовують уразливості в програмному забезпеченні.

Endpoint Protection сканує комп'ютери на наявність загроз безпеці, запобігає запуску несхвалених програм та застосовує політики брандмауера, які блокують або дозволяють мережевий трафік. Програма намагається виявити та заблокувати зловмисний трафік у корпоративній мережі або з веб-браузера, використовуючи інформацію, зібрану від користувачів для ідентифікації шкідливого програмного забезпечення. Станом на 2016 рік Symantec заявляє, що використовує дані 175 мільйонів пристроїв у 175 країнах.

1.2 Palo Alto Networks – це провідна компанія в сфері кібербезпеки, що пропонує широкий спектр рішень для захисту мереж, хмарних середовищ та кінцевих пристроїв. Вона відома своїми передовими брандмауерами нового покоління, які інтегрують функції запобігання вторгненням, захисту від шкідливих програм, контроль додатків і захист від загроз. Її продукти забезпечують глибоку видимість і контроль над трафіком, дозволяючи організаціям ефективно виявляти та реагувати на кіберзагрози.

Основними перевагами Palo Alto Networks є:

— Next-Generation Firewall – брандмауери нового покоління, які забезпечують багаторівневий захист, включаючи функції IDS/IPS, VPN, контроль додатків та аналіз загроз.

— Prisma Cloud – хмарна платформа, яка забезпечує безпеку і відповідність у багатохмарних середовищах, включаючи управління доступом, моніторинг конфігурацій і захист робочих навантажень.

— Cortex XDR – рішення для розширеного виявлення та реагування, яке використовує машинне навчання та поведінковий аналіз для виявлення складних загроз і автоматизації процесу реагування.

— WildFire – сервіс для аналізу загроз, який використовує штучний інтелект і машинне навчання для виявлення та запобігання розповсюдженню шкідливого програмного забезпечення.

— AutoFocus - інструмент для дослідження загроз, який надає інформацію про глобальні загрози та допомагає спеціалістам з безпеки швидше реагувати на інциденти.

1.3 Cisco Secure – це комплексний портфель рішень з кібербезпеки від компанії Cisco, спрямований на забезпечення всебічного захисту мереж, пристроїв і даних. Продукти та рішення Cisco Secure охоплюють різні аспекти кібербезпеки, включаючи захист кінцевих точок, мережеву безпеку, безпеку хмарних сервісів, захист електронної пошти та веб-трафіку, а також управління ідентифікацією та доступом.

До цього пакету входять наступні сервіси:

— Cisco Secure Endpoint – даний сервіс забезпечує захист кінцевих точок від шкідливих програм, вірусів, шпигунського ПЗ та інших загроз. Також він використовує розширене машинне навчання та штучний інтелект для виявлення аномалій та нових загроз. На додачу цей сервіс має централізоване управління та звітність для ефективного контролю над безпекою кінцевих точок.

— Cisco Secure Network Analytics – цей сервіс забезпечує моніторинг мережевої активності для виявлення аномальних дій та потенційних атак. Проводить аналіз мережевого трафіку в режимі реального часу для швидкого виявлення загроз. А також має інтеграцію з іншими продуктами Cisco для забезпечення комплексного захисту.

— Cisco Umbrella – це хмарний сервіс безпеки, що забезпечує захист веб-трафіку та DNS. Забезпечує безпеку від атак на основі DNS, проводить фільтрацію шкідливих веб-сайтів та блокує шкідливі домени. Даний сервіс забезпечує захист користувачів як у корпоративній мережі, так і поза нею.

— Cisco Secure Email – цей сервіс допомагає з захистом електронної пошти від фішингових атак, спаму, шкідливих вкладень та посилань. Він використовує штучний інтелект для аналізу та виявлення загроз у вхідних та вихідних повідомленнях. Цей сервіс використовує функції шифрування для забезпечення конфіденційності електронної пошти.

— Cisco Secure Access by Duo – сервіс який управляє ідентифікацією та доступом для забезпечення безпечного доступу до корпоративних ресурсів. Використовує жвофакторну аутентифікація для підвищення рівня безпеки і проводить моніторинг та управління доступом на основі політик безпеки.

— Cisco SecureX – це платформа для інтеграції та автоматизації кібербезпеки, яка об'єднує різні рішення Cisco Secure. Вона використовує інструменти для кореляції даних про загрози та автоматизації реагування на інциденти, а також центральну панель управління для спрощення моніторингу та управління безпекою.

2. Інструменти з відкритим кодом (програмне забезпечення, вихідний код якого відкритий для всіх, це означає, що будь-хто може вільно отримувати доступ, поширювати та змінювати такий софт):

2.1 Snort – це система запобігання вторгненням з відкритим вихідним кодом. Snort IPS використовує набір правил для виявлення зловмисної мережевої активності. Ці правила дозволяють ідентифікувати пакети, що відповідають певним шаблонам, і генерувати сповіщення для користувачів. Snort також можна налаштувати для блокування цих пакетів.

Основними функціями Snort є:

— Сніффер пакетів який працює як tcpdump, захоплюючи мережеві пакети в реальному часі.

— Реєстратор пакетів який записує мережевий трафік для подальшого аналізу і налагодження.

— Система запобігання вторгнень, повноцінна IPS, яка може не тільки виявляти, але і зупиняти зловмисну активність у мережі.

2.2 Suricata – це високопродуктивне програмне забезпечення для аналізу мережі та виявлення загроз із відкритим вихідним кодом, яке використовується більшістю приватних і державних організацій і впроваджується великими постачальниками для захисту своїх активів. Вона відома своєю високою продуктивністю, гнучкістю та здатністю обробляти великий обсяг трафіку у реальному часі.

Suricata легко інтегрується з мережею та може бути вбудована в численні поважні комерційні рішення та рішення з відкритим кодом.

Проект і код Suricata належать і підтримуються Фондом відкритої інформаційної безпеки (OISF), неприбутковою організацією, яка назавжди підтримує відкритий код Suricata.

До основних функцій та можливостей відноситься:

— Suricata може реєструвати запити HTTP, зберігати сертифікати TLS, видобувати файли з потоків і зберігати їх на диску..

— Suricata реєструє всі HTTP-з’єднання та запити, а також відповіді DNS для подальшого аналізу.

.

— Suricata підтримує мову підпису для відповідності відомим загрозам, порушенням політики та зловмисній поведінці.

— Suricata автоматично визначає протоколи, на будь-якому порту та застосовує відповідну логіку виявлення та журналювання.

— Suricata забезпечує розширений аналіз, дозволяючи виявляти загрози, які не можуть бути визначені за допомогою стандартного синтаксису набору правил.

Основним промисловим виходом Suricata є «Eve» це весь вихід подій і сповіщень JSON. Це забезпечує легку інтеграцію з Logstash та подібними інструментами, див. рис. 1.1.

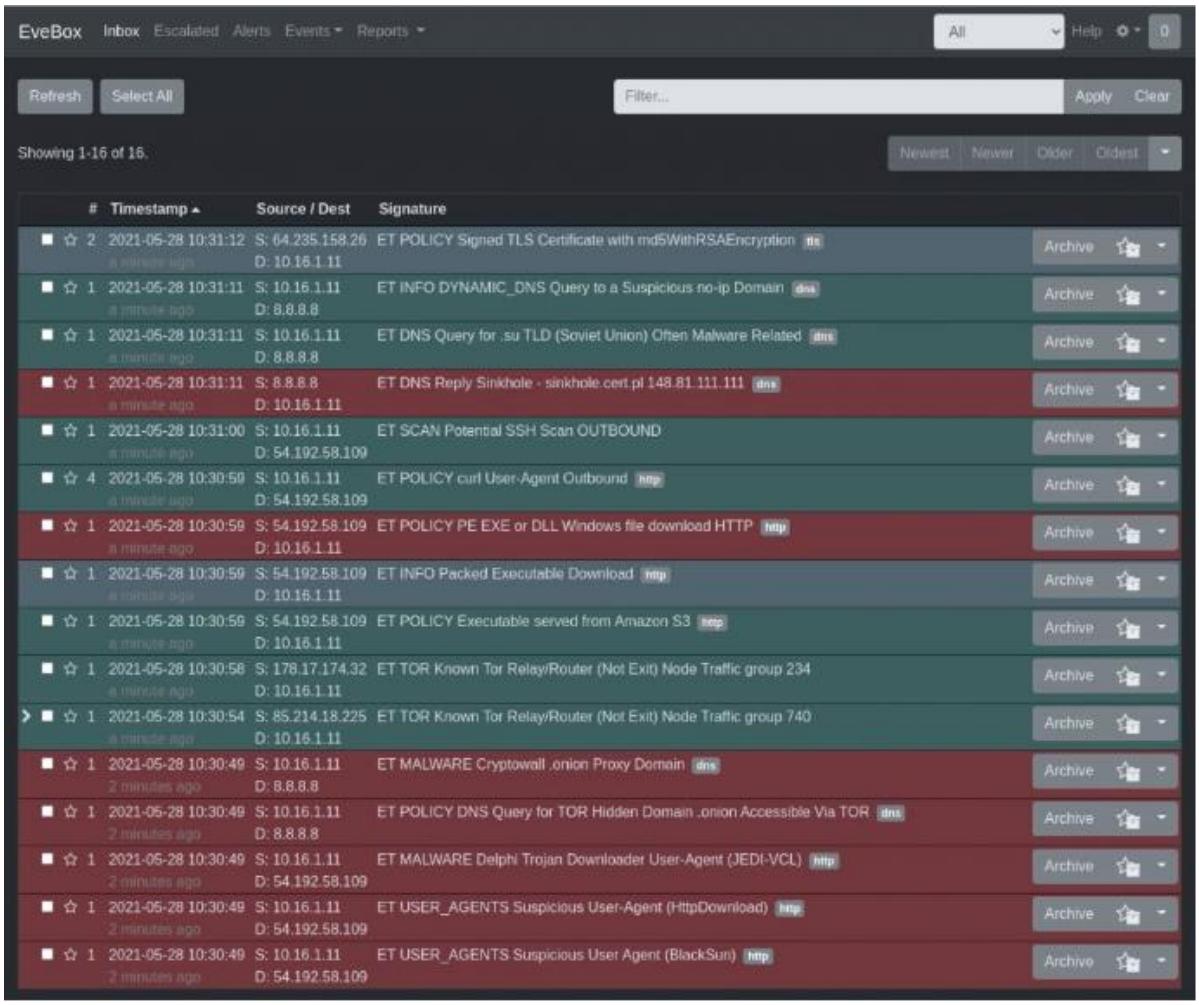


Рис. 1.1. Вікно «Eve»

2.3 Zeek – це високоефективна система аналізу мережевого трафіку, що забезпечує глибокий інтелектуальний аналіз і виявлення загроз у мережі.

Ця система може:

— Zeek аналізує мережевий трафік на глибокому рівні, розуміючи структуру і поведінку мережевих протоколів та додатків. Це дозволяє виявляти складні атаки, які можуть бути пропущені іншими засобами захисту.

— Zeek використовує потужний скриптовий мову, яка дозволяє користувачам налаштовувати його поведінку та функціональність відповідно до специфічних потреб їхньої мережі. Користувачі можуть писати свої власні сценарії для виявлення нових типів атак і аномалій.

— Zeek реєструє детальну інформацію про події в мережі, такі як HTTP-запити, FTP-сесії, SSH-з'єднання та багато іншого. Ці журнали можуть бути використані для подальшого аналізу інцидентів безпеки та аудиту.

— Zeek легко інтегрується з іншими системами кібербезпеки та інструментами аналізу даних, такими як SIEM (системи управління подіями та інформацією про безпеку). Це дозволяє створювати комплексні рішення для моніторингу та захисту мережі.

— Zeek призначений для роботи в великих мережах з високою пропускною здатністю. Його архітектура дозволяє масштабувати систему для обробки великих обсягів трафіку.

— Використовуючи методи поведінкового аналізу, Zeek здатен виявляти аномалії в мережевій активності, які можуть свідчити про наявність загроз або порушень політики безпеки.

— Zeek можна використовувати як для реального часу

моніторингу мережі, так і для офлайн аналізу збережених даних. Це робить його універсальним інструментом для різних сценаріїв використання.

На високому рівні Zeek складається з двох основних компонентів: механізму подій та інтерпретатора сценаріїв, див. рис. 1.2.

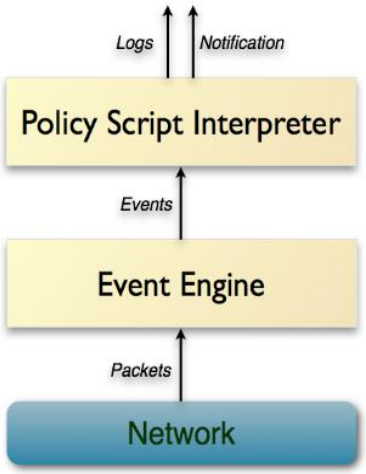


Рис. 1.2. Будова Zeek на високому рівні

Механізм подій обробляє вхідний потік пакетів, перетворюючи їх у серію подій вищого рівня, що відображають мережеву активність у нейтральних термінах.

Наприклад, кожен HTTP-запит перетворюється на подію http_request, яка містить IP-адреси, порти, URI та версію HTTP. Ця подія не містить додаткового тлумачення, такого як перевірка на відомі шкідливі URI.

Механізм подій також має архітектуру плагінів, що дозволяє розширювати функціональність Zeek шляхом додавання нових компонентів поза межами основної кодової бази.

Другий основний компонент Zeek – це інтерпретатор сценаріїв, який виконує обробники подій, написані на власній мові сценаріїв Zeek. Ці сценарії визначають політику безпеки, такі як дії, які слід виконувати при виявленні певних типів активності.

3. Рішення на базі ШІ та машинного навчання (ПЗ яке використовує алгоритми, що аналізують великі обсяги даних для виявлення аномалій та загроз):

3.1 Darktrace – це провідна компанія у сфері кібербезпеки, яка використовує штучний інтелект (ШІ) для виявлення та реагування на кіберзагрози в реальному часі.

Система Darktrace може:

— Darktrace використовує передові алгоритми машинного навчання для аналізу поведінки користувачів та пристроїв у мережі. ШІ здатний самонавчатися, що дозволяє системі адаптуватися до нових загроз без необхідності постійного оновлення сигнатур.

— Система аналізує нормальну активність у мережі та визначає відхилення від цієї норми.

— Darktrace має можливість автоматично реагувати на загрози в реальному часі. Система може ізолювати підозрілу активність, обмежуючи її вплив на мережу.

— Darktrace надає глибокий аналіз інцидентів, дозволяючи користувачам детально вивчати виявлені загрози.

— Darktrace може бути інтегрована з іншими рішеннями безпеки для підвищення ефективності загального захисту. Підтримка різних мережевих протоколів та платформ.

— Платформа забезпечує розширене моніторинг усього IT-ландшафту, включаючи хмарні сервіси, SaaS-додатки та IoT-пристрої.

Продукт Darktrace використовує неконтрольовані методи машинного навчання для побудови внутрішнього «шаблону життя» для кожної мережі, пристрою та користувача в організації. Завдяки цьому еволюційному розумінню «нормального» він може виявляти потенційні загрози, коли вони виникають у реальному часі.

3.2 Vectra AI – це компанія з кібербезпеки, яка використовує ШІ

для рішень виявлення, розслідування та реагування на гібридні атаки. Vectra AI, раніше відома як TraceVector, була заснована в 2008 році групою з 4 фахівців з кібербезпеки. Її місія полягала в тому, щоб запропонувати фахівцям із безпеки автоматизовану систему виявлення вторгнень , яка могла б протидіяти ескалації складних кібератак , кількість яких різко зросла за останні роки.

Vectra AI автоматизує виявлення загроз. Платформа Vectra AI Platform with Attack Signal Intelligence використовує AI для аналізу поведінки зловмисників і автоматичного сортування. Потім ці загрози співвідносяться, і кожному інциденту безпеки встановлюється пріоритет. Платформа використовує інтегровану систему розширеного виявлення та реагування (XDR). XDR розроблено для виявлення загроз і реагування на виклики в режимі реального часу.

Основними особливостями Vectra є:

— Vectra використовує алгоритми машинного навчання для аналізу мережевої активності та виявлення аномалій. Система самонавчається, що дозволяє їй адаптуватися до нових загроз і змін у мережевій поведінці.

— Платформа Vectra забезпечує безперервний моніторинг мережі для виявлення підозрілої активності. NDR дозволяє швидко ідентифікувати загрози та мінімізувати їх вплив.

— Система збирає та аналізує дані про активність користувачів, пристроїв та програм для створення повного контексту загроз.

— Vectra автоматизує процеси виявлення та реагування на загрози, що дозволяє зменшити навантаження на IT-відділ.

— Vectra може бути інтегрована з існуючими рішеннями для безпеки, такими як SIEM та EDR, для покращення загальної ефективності захисту.

— Платформа забезпечує аналіз загроз в реальному часі, що

дозволяє швидко реагувати на нові атаки.

— Vectra надає детальні звіти про виявлені загрози та їхній контекст.

Vectra пропонує інноваційний підхід до виявлення та реагування на кіберзагрози, використовуючи штучний інтелект для автоматизації та покращення процесів кібербезпеки. Це робить його ефективним інструментом для захисту організацій від складних та сучасних загроз.

4. Хмарні рішення (технологія, при якій обробка і зберігання інформації відбувається не на власних комп'ютерах, а в іншому місці):

4.1 Amazon GuardDuty – це служба виявлення загроз, яка безперервно моніторить, аналізує та обробляє певні джерела даних і журнали AWS у робочому середовищі. Використовуючи канали аналізу загроз, такі як списки шкідливих IP-адрес і доменів, а також моделі машинного навчання , GuardDuty здатний ідентифікувати несподівані та потенційно небезпечні дії[17].

Основними функції Amazon GuardDuty є :

— Виявлення потенційних загроз, таких як підвищення привілеїв, використання відкритих облікових даних та зв'язки із шкідливими IP-адресами і доменами.

— Проведення постійного відстеження основних джерел даних без необхідності додаткових налаштувань.

— При виявленні потенційної загрози, генерування висновків, які надають інформацію про скомпрометовані ресурси.

— Можливість інтеграції з іншими службами безпеки AWS, такими як Amazon Detective та AWS Security Hub, що в свою чергу допомагає в аналізі, дослідженні та управлінні виявленими загрозами.

4.2 Azure Security Center – це уніфікований інструмент

управління безпекою та вдосконалення захисту, який забезпечує розширений захист для середовищ Azure та гібридних хмарних середовищ.

Даний інструмент забезпечує:

— Безперервний моніторинг ресурсів Azure і гібридних середовищ для виявлення вразливостей і загроз у режимі реального часу.

— Оцінку стану безпеки ресурсів і надає рекомендації щодо вдосконалення безпеки, включаючи виправлення конфігураційних проблем і впровадження найкращих практик.

— Використання розширених алгоритмів машинного навчання та аналітику для виявлення аномальної активності та підозрілих дій, що можуть свідчити про наявність загроз або порушення безпеки.

— Комплексний захист для робочих навантажень, включаючи віртуальні машини, контейнери, бази даних та інші сервіси Azure.

— Легку інтеграцію з іншими інструментами та службами безпеки, такими як Azure Sentinel, Microsoft Defender та сторонні рішення, що забезпечує централізоване управління безпекою.

— Допомогу в управлінні доступом до ресурсів, забезпечуючи використання належних політик ідентифікації та доступу, що знижує ризик несанкціонованого доступу.

— Захист не тільки для ресурсів Azure, але й для гібридних середовищ, включаючи локальні центри обробки даних та інші хмарні платформи.

Приклад інтерфейсу Azure Security Center див. рис. 1.3.

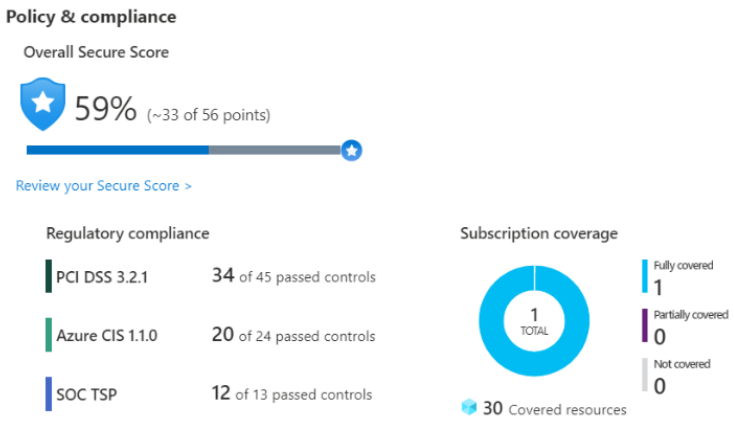


Рис 1.3. Приклад інтерфейсу

Припустімо, що ваша компанія повинна дотримуватися стандарту захисту даних індустрії платіжних карток. Цей звіт показує, що компанія має ресурси, які потребують виправлення.

У розділі "Гігієна безпеки ресурсів" ви зможете побачити стан своїх ресурсів з точки зору безпеки. Рекомендації розділені на категорії важливості: "низька", "середня" та "висока", щоб спростити пріоритизацію виправлень, див. рис. 1.4.

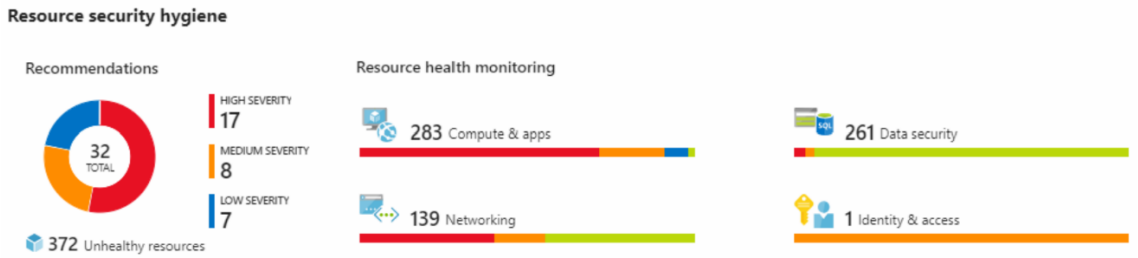


Рис. 1.4 Приклад "Гігієна безпеки ресурсів"

Оцінка безпеки (0-100%) базується на елементах контролю безпеки або групах відповідних рекомендацій щодо безпеки. Оцінка базується на відсотку контролю безпеки, який задовольняється компанією. Чим більше засобів контролю задовольняє компанія, тим вищий вона отримає.

5. Інтегровані платформи (комплексні рішення, які об'єднують різні функції безпеки в єдиній системі):

5.1 IBM QRadar – це інтегрована платформа управління інформаційною безпекою та подіями (SIEM), розроблена для виявлення, аналізу та реагування на загрози в реальному часі. QRadar збирає дані з різних джерел, аналізує їх і забезпечує комплексний огляд стану безпеки організації.

Основними можливостями платформи є:

— QRadar збирає дані з різних джерел, включаючи мережеві пристрої, сервери, додатки, бази даних і журнали подій.

— QRadar використовує механізми кореляції, щоб визначити взаємозв'язки між подіями з різних джерел і виявити потенційні загрози..

— QRadar забезпечує моніторинг безпеки в реальному часі, відстежуючи активність у мережі та додатках.

— QRadar пропонує інтуїтивно зрозумілі інформаційні панелі та звіти, які надають огляд стану безпеки організації.

— QRadar містить інструменти для детального розслідування інцидентів безпеки, включаючи аналіз сесій, трасування джерел та виявлення загроз.

— QRadar підтримує інтеграцію з іншими системами та інструментами безпеки, що забезпечує комплексний підхід до управління безпекою.

— QRadar дозволяє автоматизувати процеси реагування на інциденти, знижуючи час реакції та підвищуючи ефективність захисту.

5.2 Splunk – це платформа для аналізу та моніторингу даних, яка використовується для збору, індексації та аналізу машинно-генерованих даних у режимі реального часу. Вона забезпечує можливості пошуку, моніторингу та аналізу великих обсягів даних з різних джерел, таких як додатки, сервери, мережі, датчики тощо.

Основними можливостями Splunk є :

— Збір даних з різноманітних джерел, включаючи лог-файли, події мережі, дані додатків, дані з датчиків і багато іншого. Після цього індексація цих даних, що дозволяє швидко виконувати пошук і аналіз.

— Наявність інструментів для моніторингу інфраструктури та додатків у режимі реального часу, також можливості аналізу включають пошук аномалій, виявлення тенденцій, аналіз кореляцій та візуалізацію даних.

— Завдяки інтуїтивно зрозумілому інтерфейсу пошуку дозволяє користувачам виконувати складні запити для аналізу даних.

— Можливість генерації сповіщень на основі певних умов або подій, що допомагає виявляти та реагувати на проблеми швидше.

— Можливості автоматизації дозволяють інтегрувати Splunk з іншими системами для автоматичного реагування на певні події.

Після комплексного дослідження цих рішень можна зробити наступні висновки по програмним рішенням:

Таблиця 1.5

Висновки по програмним рішенням

Критерій	Комерційні програмні рішення	Інструменти з відкритим кодом	Рішення на базі ШІ та машинного навчання	Хмарні рішення	Інтегровані платформи
1	2	3	4	5	6
Вартість	Висока, ліцензії та підписки	Низька або безкоштовна, але потрібні ресурси на налаштування	Висока, ліцензії та інтеграція	Варіюється, часто за підпискою	Висока, але пропонує комплексні рішення

Закінчення таблиці 1.5

1	2	3	4	5	6
Масштабованість	Легко масштабується, підходить для будь-яких розмірів підприємств	Залежить від ресурсу, може вимагати додаткової підтримки	Легко масштабуєтьс я, підходить для великих організацій	Висока, масштабуєтьс я у відповідності до потреб	Висока, підходить для великих організацій
Гнучкість та налаштування я	Обмежена налаштуван нями, передбачени ми виробником	Висока гнучкість та налаштування під конкретні потреби	Висока, налаштовуєть ся під потреби підприємства	Залежить від провайдера, обмежена можливість налаштування	Висока, пропонує модульні підходи та налаштуван ня
Інтеграція з іншими системами	Зазвичай добре інтегрується з іншими продуктами виробника	Може потребувати додаткової роботи для інтеграції	Інтеграція з існуючими інфраструкту рами кібербезпеки	Легко інтегрується з хмарними службами та іншими продуктами	Висока інтеграція з різними системами безпеки
Простота впровадже ння	Вимагає технічної підтримки та часу на налаштуван ня	Може бути складною для впровадження без достатньої технічної підготовки	Висока, але потрібні знання у сфері ШІ	Висока, швидке розгортання та налаштування	Вимагає серйозного технічного обслуговува ння та впровадже ня

Після дослідження даного питання можна зробити висноки, що комерційні програмні рішення надають надійний захист та масштабованість, але часто вимагають значних фінансових ресурсів. Інструменти з відкритим кодом дозволяють адаптувати та налаштовувати систему під конкретні потреби, проте

їхня ефективність може залежати від досвіду користувачів. Рішення на базі штучного інтелекту та машинного навчання забезпечують автоматичне виявлення складних та нових загроз, але їх впровадження та налаштування можуть бути складними. Хмарні рішення відзначаються високою доступністю та інтеграцією, проте викликають певні занепокоєння щодо безпеки даних. Інтегровані платформи забезпечують комплексний підхід до безпеки, але можуть бути надто складними для малих організацій.

Враховуючи ці аспекти, стає очевидним, що для ефективного виявлення кібератак потрібен комплексний підхід, який поєднує в собі переваги кількох класів рішень. Таким чином, для подальшого вдосконалення систем виявлення кібератак пропонується розробка програмного рішення, яке інтегрує найкращі практики з кожного з аналізованих класів. Зокрема, варто звернути увагу на використання штучного інтелекту для автоматизації виявлення загроз, що дозволить підвищити ефективність реагування на нові та складні атаки.

1.4 Висновки до розділу 1

У Розділі 1 ми проаналізували сучасні методи виявлення кібератак, класифікували основні типи кібератак та дослідили актуальні програмні рішення для їх виявлення. Було розглянуто як традиційні, так і новітні підходи, що базуються на машинному навчанні та штучному інтелекті. Ми також проаналізували переваги та обмеження різних інструментів, включаючи комерційні рішення, інструменти з відкритим кодом, хмарні сервіси та інтегровані платформи.

На основі проведеного аналізу, можна зробити висновок, що хоча традиційні методи залишаються важливими, зростаюча складність кібератак вимагає більш інноваційних підходів. Новітні рішення, що використовують ШІ та машинне навчання, показали свою ефективність у виявленні складних та нових загроз. Однак, кожен клас інструментів має свої специфічні переваги та

недоліки, що підкреслює необхідність комбінування різних технологій для забезпечення максимальної безпеки.

У наступному розділі ми перейдемо до розробки архітектури програмного рішення, яке буде враховувати всі виявлені під час аналізу фактори. Це дозволить створити більш гнучке та ефективне рішення для виявлення кібератак, яке можна буде інтегрувати в сучасні системи кібербезпеки.

РОЗДІЛ 2.
ТЕОРЕТИЧНИЙ ОПИС ТА АРХІТЕКТУРА ПРОГРАМНОГО МОДУЛЯ

Швидкий розвиток комунікаційних технологій значно прискорює прогрес у сфері інформаційних технологій, змінюючи наш спосіб життя, роботи та спілкування. Розумні пристрої та системи стають невід'ємною частиною нашого повсякденного життя, що робить питання кібербезпеки надзвичайно актуальним.

2.1 Аналіз проблеми та обґрунтування вибору програмного рішення

Традиційні методи виявлення загроз, такі як сигнатурний аналіз та ручне моніторинг, вже не відповідають вимогам часу через їхню обмежену здатність реагувати на нові, невідомі типи атак. Це породжує необхідність пошуку нових підходів та рішень, які б забезпечили ефективний захист інформації від кібератакюр[18].

Штучний інтелект (ШІ) та машинне навчання (МН) пропонують такі можливості завдяки використанню складних математичних моделей, які здатні виявляти невідповідності в поведінці системи та прогнозувати потенційні загрози на основі історичних даних[19].

Для обґрунтування вибору технологій штучного інтелекту та машинного навчання було розглянуто базові математичні моделі, що ілюструють ефективність таких підходів у порівнянні з традиційними методами.

1. Оцінка ймовірності виявлення загрози (P_D):

$$\boldsymbol{P_D = 1 - \prod_{i=1}^{n}(1 - p_i)}, \tag{2.1}$$

де $\boldsymbol{p_i}$ – ймовірність виявлення загрози за допомогою і-го методу або моделі. Це дозволяє показати, що поєднання кількох моделей або методів виявлення підвищує загальну ймовірність виявлення загрози.

Дана формула, відома як формула для обчислення ймовірності виявлення загрози. Вона базується на основах теорії ймовірностей і часто використовується в контексті комбінаторного аналізу незалежних подій.

Основними аксіомами, що лежать в основі формули оцінки ймовірності виявлення загроз є:

1. Якщо ймовірність певної події становить $\boldsymbol{p_i}$, то ймовірність того, що ця подія не відбудеться, дорівнює $\mathbf{1} - \boldsymbol{p_i}$.
2. Якщо у вас є кілька незалежних подій, ймовірність того, що жодна з них не відбудеться, дорівнює добутку ймовірностей того, що кожна подія не відбудеться. Якщо у вас є n подій, то ймовірність того, що жодна з цих подій не відбудеться, буде дорівнювати $\prod_{i=1}^{n}(\mathbf{1} - \boldsymbol{p_i})$
3. Відповідно, загальна ймовірність виявлення хоча б однієї загрози в системі визначається як доповнення до ймовірності невиявлення жодної загрози:

$$\boldsymbol{P_D} = \mathbf{1} - \prod_{i=1}^{n}(\mathbf{1} - \boldsymbol{p_i})$$

2. Модель оцінки адаптивності атак ($A(t)$):

Для моделювання адаптивності атак було використано функцію росту, яка описує ймовірність того, що атакувальник зможе уникнути виявлення:

$$\boldsymbol{A(t)} = \frac{\boldsymbol{A_0}}{\mathbf{1} + \frac{\boldsymbol{A_0}}{\boldsymbol{K}}\boldsymbol{e^{-rt}}}, \qquad (2.2)$$

де $\boldsymbol{A_0}$ це значення адаптивності атак на початковому етапі. Воно описує, з якої точки зору починається процес адаптації атак.

$\boldsymbol{K}$ визначає максимальний рівень, до якого можуть адаптуватися атаки. Ця межа обмежена ресурсами зловмисників або можливостями протидіючих захисних заходів.

$\boldsymbol{r}$ це параметр, який визначає, як швидко атаки стають адаптивнішими. Чим вищий $\boldsymbol{r}$, тим швидше атаки пристосовуються до нових умов.

t відображає момент часу, на який оцінюється рівень адаптивності атак.

Основою для цієї моделі є логістична функція, яка широко використовується для опису процесів зростання, обмежених деяким насиченням або максимально можливим рівнем. У даному контексті модель оцінює адаптивність атак із плином часу, враховуючи те, що атаки не можуть зростати безмежно та мають межу ефективності.

Це може включати зміну тактики атакуючих, які стають дедалі витонченішими та здатними обходити існуючі захисні механізми.

Ця модель буде використовуватися для прогнозування, як атаки можуть змінюватися з часом, і для планування відповідних заходів захисту. На основі даних про попередні атаки та їх еволюцію ми можемо налаштувати параметри моделі A_0, K, і r для прогнозування майбутніх змін у тактиці зловмисників. Розуміння того, як змінюється адаптивність атак, дозволить оптимізувати наявні механізми захисту, забезпечуючи швидку реакцію на нові загрози.

Ця модель допоможе візуалізувати і передбачати майбутні загрози на основі адаптивності атак, що дозволить приймати обґрунтовані рішення щодо вдосконалення кіберзахисту.

3. Формула для оцінки навантаження на систему (L):

$$L = \sum_{i-1}^{n} \frac{S_i \times T_i}{R_i}, \tag{2.3}$$

де n – це загальна кількість операцій або завдань, які система має виконати. Це може включати аналіз пакетів даних, виявлення аномалій, аналіз журналів подій тощо.

S_i представляє розмір і-ої операції або обсягу даних, які необхідно обробити. Чим більше даних обробляється, тим вище буде значення S_i.

T_i – це час, необхідний для обробки і-ої операції. Цей показник враховує складність обробки та ресурси, необхідні для завершення завдання.

$\boldsymbol{R_i}$ відображає кількість ресурсів , що виділені для виконання і-ої операції. Більше доступних ресурсів може зменшити загальне навантаження на систему.

Формула оцінки навантаження на систему базується на принципах розподілу обчислювальних ресурсів відповідно до завдань, які виконує система в режимі реального часу. Ця модель дозволяє оцінити загальний рівень навантаження, враховуючи ресурси, необхідні для обробки різних типів даних або завдань.

Параметр $\boldsymbol{L}$ представляє загальне навантаження на систему, яке виникає внаслідок виконання різних завдань під час обробки даних. Ця величина допомагає оцінити, наскільки завантажена система в конкретний момент часу.

Дана формула дозволяє оцінити, наскільки завантажена система кібербезпеки під час виконання різних завдань. Це важливо для розуміння, чи зможе система ефективно справлятися зі збільшенням обсягів роботи та чи необхідно додатково оптимізувати процеси або виділити більше ресурсів.

Аналіз навантаження на систему дозволяє ідентифікувати «вузькі місця» в продуктивності, що потребують додаткових ресурсів або оптимізації алгоритмів. Знаючи значення L, можна прогнозувати необхідність у масштабуванні системи, особливо при збільшенні обсягів даних або кількості атак. В свою чергу підвищене значення L може сигналізувати про можливі проблеми в системі, такі як недостатність ресурсів або неефективність існуючих процесів.

Загалом у даному розділі буде проведено глибокий аналіз проблем, з якими стикаються сучасні системи виявлення кібератак. Крім того, буде розглянуто, чому саме технології штучного інтелекту та машинного навчання можуть стати ефективним рішенням для подолання цих викликів, та обґрунтовано вибір програмного рішення, заснованого на цих технологіях.

2.1.1 Опис проблеми та мотивації вибору програмного рішення

Сучасні системи кібербезпеки стикаються з постійно зростаючим обсягом загроз, що швидко еволюціонують. Зокрема, зі збільшенням складності та частоти кібератак, існуючі методи захисту виявляються недостатньо ефективними. Традиційні системи виявлення загроз здебільшого базуються на статичних правилах та сигнатурах, що робить їх вразливими до нових, невідомих видів атак[20].

Однією з основних проблем є те, що кібератаки стають дедалі більш адаптивними та витонченими. Зловмисники використовують технології, що дозволяють їм швидко змінювати свою поведінку, ухилятися від виявлення та знаходити нові способи проникнення в системи. Це робить виявлення загроз складним і вимагає нових підходів до аналізу та реагування.

Крім того, збільшення обсягів мережевого трафіку та даних ускладнює процес моніторингу та виявлення аномалій. Сучасні системи часто не в змозі обробляти такі великі обсяги інформації в реальному часі, що призводить до затримок у виявленні загроз і зниження ефективності захисту.

Таким чином, виникає нагальна потреба в нових підходах, що здатні адаптуватися до мінливих умов і забезпечувати більш гнучке та ефективне виявлення кібератак.

2.1.2 Виклики перед сучасними інструментами виявлення кібератак.

Щодня зловмисники вдосконалюють свої методи атак, використовуючи нові техніки, які можуть обходити традиційні засоби захисту. Наприклад, використання шифрування та стеганографії для приховування зловмисного коду ускладнює виявлення загроз за допомогою класичних сигнатурних підходів.

З кожним днем збільшується кількості мережевого трафіку, користувачів та пристроїв, що в свою чергу ускладнює аналіз даних у реальному часі.

Інструменти виявлення часто не встигають обробляти такі великі обсяги даних, що може призводити до пропуску критично важливих загроз.

Хоч кіберзагрози стають дедалі складнішими та хитрішими, маскуючись під легітимні процеси, багато інструментів для виявлення загроз працюють ізольовано, не забезпечуючи достатньої інтеграції між собою та іншими системами безпеки. Це в свою чергу призводить до фрагментації даних та ускладнює їхній аналіз у контексті загальної картини.

В теперішніх умовах, коли атаки можуть розвиватися дуже швидко, критично важливо, щоб інструменти виявлення могли вчасно реагувати на загрози. Проте, через велику кількість хибних спрацювань, а також затримки в аналізі та прийнятті рішень, багато інструментів не встигають належним чином зреагувати на загрозу до того, як вона завдасть шкоди[21].

2.1.3 Неповноцінність традиційних методи виявлення кібератак

Традиційні методи, такі як сигнатурний аналіз, можуть виявляти атаки лише після того, як вони вже відбулися або коли їхні ознаки були внесені в бази даних. Це призводить до того, що нові або змінені атаки можуть залишатися невиявленими протягом тривалого часу.

Також методи, засновані на статичних правилах, часто генерують багато хибнопозитивних сповіщень, що ускладнює роботу аналітиків і збільшує ризик пропустити реальні загрози через "шум" великої кількості попереджень.

Ці методи не ефективні проти нових, раніше невідомих загроз. Оскільки вони залежать від відомих сигнатур та правил, вони не можуть виявити нові атаки, які ще не були задокументовані. Звичайні підходи важко адаптуються до складних і динамічних середовищ, де атаки постійно еволюціонують. Зміни у структурі мережі, застосуванні нових технологій або поведінці користувачів можуть ускладнювати ефективність статичних методів.

Традиційні методи часто не враховують повний контекст атак, таких як поведінкові зміни в системі або взаємодія між різними компонентами мережі. Це

обмежує здатність виявляти складні та багатоетапні атаки. Врешті решт можна зробити висновок, що традиційні методи не справляються з викликами сучасного кіберсередовища, що в свою чергу підкреслює необхідність нових підходів, таких як рішення на базі штучного інтелекту та машинного навчання.

2.1.4 Особливості технології на основі штучного інтелекту та машинного навчання

Ця технологія здатна аналізувати великий обсяг даних у режимі реального часу, що дозволяє їй ефективно виявляти нові, раніше невідомі загрози. Завдяки адаптивним алгоритмам, система може навчатися на нових даних і покращувати свої моделі виявлення з часом. За допомогою цих алгоритмів зменшується кількість хибнопозитивних сповіщень, що в свою чергу значно підвищує ефективність роботи аналітиків безпеки.

Обрана технологія дозволяє реалізувати комплексний підхід, поєднуючи різні аспекти кіберзахисту в єдину систему. Завдяки такому підходу можна об'єднувати дані з різних джерел і аналізувати їх у комплексі, що забезпечує більш глибоке розуміння подій у мережі. Це допомагає виявляти складні та багатоступеневі атаки, які можуть залишатися непоміченими при використанні традиційних методів. Також цей підхід передбачає, що технологія буде ефективно працювати в тандемі з іншими вже впровадженими рішеннями, що підвищує загальну стійкість системи до кіберзагроз.

Вибрана технологія забезпечує значні переваги в сучасному кіберзахисті завдяки своїй здатності виявляти нові загрози та зменшувати хибнопозитивні сповіщення. Комплексний підхід цієї технології дозволяє інтегрувати та аналізувати дані з різних джерел, що забезпечує більш точне та ефективне виявлення складних атак. Це також забезпечує гармонійне доповнення до існуючих рішень, підвищуючи загальну стійкість системи до кіберзагроз. Усе це робить технологію потужним інструментом для забезпечення надійного захисту інформаційних систем.

2.2 Загальний опис архітектури

У цьому розділі ми розглянемо загальну архітектуру програмного рішення, яке буде розроблене в рамках даної кваліфікаційної роботи. Основна мета цього рішення - забезпечити ефективне виявлення та реагування на кіберзагрози за допомогою технологій штучного інтелекту та машинного навчання в комп'ютерних системах. Архітектура системи складається з декількох ключових модулів, кожен з яких виконує важливу роль у загальному процесі захисту від кібератак. У цьому розділі буде представлено огляд основних компонентів системи, їх взаємодію та функціональні можливості, що забезпечують комплексний підхід до аналізу даних і виявлення загроз[22].

2.2.1 Загальна структура системи та вимоги до неї

Структура системи складається з наступних компонентів:

1. Модуль збору даних функція якого полягає в зборі даних з різних джерел, таких як мережевий трафік, журнали подій, та інші релевантні ресурси. Цей модуль отримує дані та передає їх на обробку в модуль попередньої обробки.
2. Модуль попередньої обробки даних функція якого полягає в очищенні, нормалізації та підготовці даних для аналізу. Модуль також відповідає за видалення шуму та перетворення даних у формат, зручний для моделювання. Він відповідає за отримання даних від модуля збору даних і передачу їх до модуля аналізу.
3. Модуль аналізу на основі ШІ функція якого полягає в застосуванні моделей машинного навчання для аналізу даних у реальному часі з метою виявлення аномалій або потенційних загроз. Він працює використовуючи підготовлені дані з модуля попередньої обробки та проводить їх аналіз на предмет підозрілих дій чи аномалій.

4. Модуль реагування функція якого полягає в генерації відповідних дій або сповіщень на основі результатів аналізу. Наприклад, сповіщення про виявлені загрози або автоматичне блокування підозрілої активності. Він отримує результати від модуля аналізу та виконує відповідні дії.

5. Модуль зберігання даних функція якого полягає в збереженні як сирих даних, так і результатів аналізу для подальшого використання чи звітності. Цей модуль взаємодіє з усіма іншими модулями для запису та зберігання даних.

Дана архітектура була розроблена для того щоб показати основні можливості аналізу даних на основі ШІ. У майбутньому дане рішення можна буде розширити, додавши нові функції та модулі для інтеграції з іншими системами.

Після визначення основної архітектури системи важливо оцінити ресурси, необхідні для її ефективної роботи. Оскільки система буде обробляти дані в реальному часі, аналізувати загрози та приймати рішення на основі алгоритмів машинного навчання, важливо передбачити вимоги до обчислювальних ресурсів, зокрема процесорного часу, пам'яті, та пропускної здатності системи. У цьому контексті було розглянуто кілька ключових формул, що дозволять здійснити попередні розрахунки та прогнозування навантаження на систему:

1. Оцінка обчислювальних ресурсів (T_{CPU}):

$$\boldsymbol{T_{CPU}} = \frac{\boldsymbol{N + CPI}}{\boldsymbol{F_{clock}}}, \tag{2.4}$$

де $\boldsymbol{N}$ – загальна кількість інструкцій, які система має виконати. Це може включати аналіз пакетів даних, виявлення аномалій, аналіз журналів подій тощо.

$\boldsymbol{CPI}$ представляє кількість тактів процесора, необхідних для виконання однієї інструкції. Чим вище складність інструкцій, тим більше тактів буде потрібно.

$\boldsymbol{F_{clock}}$ – тактова частота процесора, що відображає швидкість, з якою процесор може виконувати інструкції. Вища тактова частота дозволяє зменшити загальний час обробки.

Формула оцінки обчислювальних ресурсів базується на принципах розподілу обчислювальних ресурсів відповідно до завдань, які виконує система в режимі реального часу. Ця модель дозволяє оцінити час, необхідний для виконання обчислень на процесорі, враховуючи кількість інструкцій та їх складність.

Параметр $\boldsymbol{T_{CPU}}$ представляє час, необхідний для обробки інструкцій процесором. Ця величина допомагає оцінити, наскільки швидко система може обробити дані в конкретний момент часу.

Ця формула допоможе оцінити, скільки часу потрібно для обробки даних на процесорі при виконанні алгоритмів машинного навчання або інших обчислювальних задач, також вона дозволить визначити, які вимоги до процесора повинні бути для підтримки необхідної продуктивності системи.

Оцінка обчислювальних ресурсів допоможе в аналізі продуктивності системи на різних етапах і дозволить зрозуміти, як зміна кількості оброблюваних інструкцій або тактової частоти процесора вплине на загальний час обробки даних. Використовуючи цю формулу, можна оптимізувати алгоритми машинного навчання, зменшуючи $\boldsymbol{CPI}$ або підвищуючи $\boldsymbol{F_{clock}}$ для зменшення часу обробки і підвищення ефективності системи.

2. Оцінка використання пам'яті(M):

$$\boldsymbol{M = \sum_{i=1}^{n} (D_i \times S_i)}, \tag{2.5}$$

де $\boldsymbol{M}$ – загальне використання пам'яті. Ця величина допомагає оцінити загальний обсяг пам'яті, який використовується для виконання всіх завдань у системі. Це критично важливо для забезпечення стабільної роботи системи кібербезпеки.

$\boldsymbol{n}$ – загальна кількість операцій або завдань, які система має виконати. Це може включати аналіз пакетів даних, виявлення аномалій, аналіз журналів подій тощо. Різні операції можуть вимагати різної кількості пам'яті, залежно від обсягу та складності даних.

$\boldsymbol{D_i}$ – кількість даних, які необхідно обробити в рамках i-ї операції. Це обсяг даних, який необхідно обробити в рамках конкретної операції. Наприклад, аналіз мережевих пакетів або журналів подій може вимагати обробки великої кількості даних.

$\boldsymbol{S_i}$ – розмір даних i-ї операції. Це розмір даних, який потрібно обробити для конкретної операції. Величина може варіюватися в залежності від типу даних і складності обробки.

Основою для цієї моделі є класична формула для оцінки використання пам'яті, яка враховує кількість даних і їхній розмір, необхідні для обробки різних завдань у системі. Подібна модель часто використовується для оцінки ресурсів у системах обробки даних.

Формула $\boldsymbol{M = \sum_{i=1}^{n}(D_i \times S_i)}$ дозволяє оцінити загальний обсяг пам'яті, необхідний для виконання всіх завдань у системі. Це важливо для розуміння, скільки пам'яті потрібно для обробки даних і виявлення аномалій у реальному часі.

Ця модель допомагає оцінити необхідний обсяг пам'яті для виконання різних завдань у системі кібербезпеки. Це дозволяє ідентифікувати «вузькі місця» у використанні пам'яті та оптимізувати процеси для забезпечення ефективної роботи системи.

Формула $\boldsymbol{M = \sum_{i=1}^{n}(D_i \times S_i)}$ дозволяє:

— Оцінити, чи вистачає наявних ресурсів пам'яті для обробки всіх завдань у системі.

— Планувати додаткові ресурси, якщо поточних недостатньо.

— Виявляти та усувати потенційні проблеми з продуктивністю, що виникають через нестачу пам'яті.

Ця модель допомагає прогнозувати потреби у пам'яті при зростанні обсягу даних або кількості атак, що дозволяє вчасно масштабувати систему та забезпечити її стабільну роботу.

3. Оцінка пропускної здатності системи(TP):

$$\boldsymbol{TP} = \frac{\boldsymbol{N}}{\boldsymbol{T_{total}}}, \tag{2.6}$$

де $\boldsymbol{TP}$ – пропускна здатність системи. Ця величина визначає, скільки запитів або операцій система може обробити за певний проміжок часу. Це критично важливо для забезпечення високої продуктивності системи кібербезпеки, особливо під час активних атак.

$\boldsymbol{N}$ – кількість оброблених запитів або операцій. Це може включати аналіз мережевих пакетів, виявлення аномалій, аналіз журналів подій тощо. Чим більше запитів система може обробити, тим вищою є її пропускна здатність.

$\boldsymbol{T_{total}}$ – загальний час, витрачений на обробку цих запитів або операцій. Це час, необхідний для обробки всіх запитів або операцій. Враховує весь процес обробки, від отримання даних до завершення аналізу.

Основою для цієї моделі є класична формула для оцінки пропускної здатності систем, яка використовується для вимірювання продуктивності в обчислювальних системах. Подібна модель широко використовується в комп'ютерних науках для оцінки ефективності роботи системи.

Формула $\boldsymbol{TP} = \frac{\boldsymbol{N}}{\boldsymbol{T_{total}}}$ допомагає оцінити пропускну здатність системи кібербезпеки. Це дозволяє визначити, чи може система ефективно обробляти запити в умовах підвищеного навантаження, та планувати необхідні ресурси для підтримки продуктивності.

Ця модель дозволяє оцінити, наскільки продуктивною є система кібербезпеки під час обробки різних запитів та операцій. Знаючи значення $\boldsymbol{TP}$, можна:

— Прогнозувати потреби у масштабуванні системи при збільшенні обсягів даних або кількості атак.

— Ідентифікувати проблеми з продуктивністю та вживати заходів для їх усунення.

— Забезпечити стабільну роботу системи під час активних атак та підвищеного навантаження.

Таким чином, формула $\boldsymbol{TP = \frac{N}{T_{total}}}$ є адаптованою версією класичної моделі оцінки пропускної здатності системи, що дозволяє оцінити продуктивність та ефективність роботи системи кібербезпеки у системі.

Після розрахунку та оптимізації ресурсів, необхідних для роботи системи, важливо визначити функціональні та нефункціональні вимоги. Функціональні вимоги описують основні задачі, які система повинна виконувати, щоб досягти своїх цілей, тоді як нефункціональні вимоги стосуються якості та умов, за яких ці функції повинні бути виконані. Визначення цих вимог дозволяє встановити чіткі критерії для розробки, тестування та впровадження системи, забезпечуючи її ефективність, надійність та відповідність потребам користувачів.

Функціональні вимоги включають конкретні завдання, які система повинна виконувати для виявлення та реагування на загрози у комп'ютерній мережі, а також механізми для автоматичного реагування та інтеграції в існуючу інфраструктуру.

Нефункціональні вимоги визначають вимоги до продуктивності, надійності, безпеки та зручності використання системи. Вони забезпечують умови, за яких система повинна функціонувати, та встановлюють стандарти для її експлуатації.

Таким чином, на основі оцінки необхідних ресурсів та розрахунків, наведених вище, ми можемо сформулювати функціональні та нефункціональні вимоги до системи, які забезпечать її ефективну та надійну роботу.

Функціональні вимоги до системи включають:

1. Система повинна використовувати алгоритми штучного інтелекту та машинного навчання для виявлення загроз у комп'ютерній мережі. Це включає аналіз обмеженого набору даних

для виявлення основних типів загроз, таких як аномалії у трафіку або підозріла активність.

2. Система повинна забезпечувати механізми для автоматичного реагування на виявлені загрози, включаючи сповіщення користувача та запис подій у лог-файл.

3. Система повинна бути здатна легко інтегруватися в існуючу інфраструктуру організації без необхідності значних змін або адаптацій.

4. Архітектура системи повинна дозволяти легке розширення функціоналу в майбутньому. Наприклад, можливість додавання нових моделей для аналізу даних або інтеграції з іншими системами безпеки. Також має бути передбачена можливість оновлення алгоритмів і моделей машинного навчання для покращення точності та ефективності виявлення загроз.

5. Система повинна мати простий і зрозумілий інтерфейс для взємодії з користувачем, який дозволяє отримувати основні результати аналізу, переглядати звіти та отримувати сповіщення про загрози.

Нефункціональні вимоги до системи включають:

1. Система повинна забезпечувати обробку даних у реальному часі або близько до реального часу, щоб своєчасно виявляти та реагувати на загрози.

2. Система повинна мати високий рівень надійності, з мінімальними перервами в роботі, а також можливість відновлення після збоїв.

3. Система повинна мати можливість розширення для підтримки збільшених обсягів даних і додавання нових функцій без втрати продуктивності.

4. Система повинна забезпечувати захист даних, які обробляються, а також контроль доступу до різних компонентів системи. Це включає шифрування даних, автентифікацію та авторизацію користувачів.

5. Система повинна бути спроектована так, щоб її можна було легко підтримувати та оновлювати з мінімальними затратами на адміністрування та обслуговування.

Завдяки цим вимогам буде забезпечено баланс між функціональністю та простотою, що дозволить зосередитися на демонстрації основного потенціалу системи.

2.2.2 Архітектура даних

Архітектура даних складається з моделей, політик, правил або стандартів, які визначають, які дані збираються, і як вони зберігаються, розміщуються, інтегруються і використовуються для використання в системах даних і в організаціях. Дані зазвичай є одним із декількох доменів архітектури, які складають основу архітектури підприємства або архітектури рішення.

Для забезпечення ефективної роботи системи виявлення та реагування на кіберзагрози будуть використовуватися різноманітні типи даних. Це включає дані з мережевих журналів (логів), телеметрію, журнали доступу, дані про поведінку користувачів, та інші дані, пов'язані з безпекою. Джерела цих даних можуть бути наступними:

— Журнали подій – логи операційної системи, мережевих пристроїв, серверів і додатків.

— Мережевий трафік – дані, зібрані з мережевих інтерфейсів або спеціалізованих пристроїв.

— Антивірусне ПЗ та системи захисту кінцевих точок – дані від різних інструментів безпеки, встановлених на комп'ютерах користувачів.

— Зовнішні джерела – чорні списки IP-адрес, бази даних загроз та інші ресурси, що можуть бути корисними для виявлення нових загроз.

Збір даних є ключовим етапом у функціонуванні системи виявлення та реагування на кіберзагрози. Основна мета цього етапу – отримати необхідну інформацію для аналізу, на основі якої система зможе виявляти загрози та приймати рішення щодо реагування.

Збір даних мережевого трафіку на рівні мережевих інтерфейсів або спеціалізованих пристроїв. Це дасть можливість аналізувати поведінку мережі і виявляти аномальні патерни, які можуть бути ознаками кіберзагроз.

Система може отримувати дані від антивірусного ПЗ, систем захисту кінцевих точок, а також інших інструментів безпеки, встановлених на комп'ютерах користувачів.

Включення зовнішніх джерел даних, таких як чорні списки IP-адрес, бази даних загроз та інші ресурси, що можуть бути корисними для виявлення нових загроз.

Процес збору даних відбуватиметься за допомогою встановлення спеціальних програмних агентів на кінцевих пристроях або мережевих пристроях для збору та передачі даних на центральний сервер для аналізу, також за допомогою використання методів пасивного моніторингу мережевого трафіку без встановлення агентів, що може зменшити навантаження на інфраструктуру. Залежно від критичності даних та ресурсів, збір може відбуватися в реальному часі або періодично (наприклад, щогодини або щодня).

Щодо зберігання даних буде передбачено 2 способи:

— Локальне зберігання - зібрані дані можуть зберігатися локально на сервері для швидкого доступу та аналізу. Це забезпечить мінімальні затримки в обробці.

— Хмарне зберігання - у випадках, коли потрібен доступ до великих обсягів даних або зберігання історичних даних, можна буде використовувати хмарні рішення для забезпечення масштабованості та доступності.

З допомогою цього етапу система забезпечується всіма необхідними даними для аналізу загроз. Цей етап є основою для побудови ефективної системи кібербезпеки.

Основні компоненти архітектури даних включають модулі збору, обробки, зберігання даних, навчання моделі та реакції на загрози.

1. Модуль збору даних

Цей модуль відповідає за збір даних з різноманітних джерел, таких як мережевий трафік, журнали подій, поведінкові дані, та інші релевантні джерела. Важливо забезпечити різноманітність і якість зібраних даних, щоб покращити навчання моделей. Модуль збору даних має бути інтегрований з іншими частинами системи, що дозволить безперебійно передавати зібрані дані до наступних етапів обробки.

2. Модуль попередньої обробки даних

Цей модуль відповідає за очищення зібраних даних від шуму, дублікатів і непотрібних елементів. Якість даних має вирішальне значення для ефективного навчання моделей. Дані повинні бути нормалізовані та перетворені у формат, зручний для подальшого аналізу і моделювання. Це включає, наприклад, масштабування числових значень або категоризацію текстових даних

3. Модуль зберігання даних

Цей модуль відповідає за надійне зберігання як сирих, так і оброблених даних, забезпечуючи можливість їх швидкого доступу під час аналізу та навчання моделей. Дані мають бути організовані у структурованій формі, що полегшує їх використання. Забезпечення конфіденційності та цілісності даних під час їх зберігання є критичним завданням. Модуль зберігання повинен включати механізми шифрування та контролю доступу для захисту даних від несанкціонованого доступу.

4. Модуль навчання моделі

Після попередньої обробки дані передаються до модуля навчання. Тут проводиться поділ на навчальну та тестову вибірки, а також

оптимізація параметрів моделей машинного навчання. Модуль навчання відповідає за процес навчання моделей, підвищення їх точності через оптимізацію гіперпараметрів, і регулярне оновлення моделей на основі нових даних.

5. Модуль аналізу та реакції

Після навчання моделей, модуль аналізу здійснює обробку нових даних у реальному часі, з метою виявлення аномалій, загроз або інших підозрілих активностей. На основі результатів аналізу, модуль реакції виконує певні дії, такі як створення сповіщень, запуск автоматичних сценаріїв реагування, або запис результатів для подальшого аналізу.

6. Модуль оновлення моделі

Модуль постійно моніторить ефективність моделей на нових даних та проводить регулярні оновлення, щоб забезпечити актуальність та точність моделей. Оновлення моделі включає повторне навчання з використанням нових даних та адаптацію до змінюваних умов, що дозволяє зберігати високу точність виявлення загроз.

На схемі нижче показано візуалізований процес збору даних та показано, як різні компоненти системи взаємодіють між собою, також було підкреслено основні етапи збору, обробки, та зберігання даних, див. рис. 2.1.

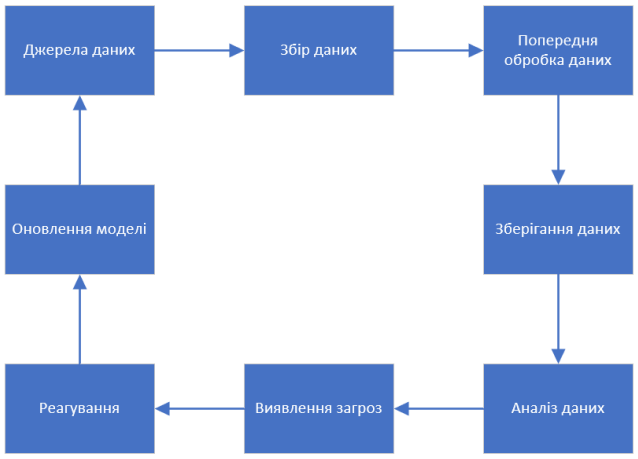


Рис. 2.1. Архітектура програмного модуля

1. Джерела даних. На цьому етапі система отримує дані з різних джерел. Це можуть бути мережеві журнали (логи), телеметрія, журнали доступу, дані про поведінку користувачів, та інші дані, пов'язані з безпекою. Сирі дані передаються до етапу збору даних (Data Ingestion) для подальшої обробки.

2. Збір даних. На цьому етапі дані збираються та об'єднуються в єдиному форматі. Можливо, також відбувається первинна фільтрація, щоб видалити непотрібні або дублікатні дані. Оброблені та відфільтровані дані передаються на етап попередньої обробки.

3. Попередня обробка даних. Дані проходять процес очищення та нормалізації. Це включає видалення некоректних або нерелевантних даних, приведення даних до єдиного формату, а також витягування необхідних ознак (features), які будуть використовуватися в процесі аналізу. Очищені та нормалізовані дані передаються до сховища даних для зберігання.

4. Зберігання даних. На цьому етапі дані зберігаються в структурованому вигляді, готовому для подальшого аналізу. Зазвичай це може бути реляційна база даних або сховище даних великого обсягу, що підтримує швидкий доступ до інформації. Збережені дані передаються на етап аналізу.

5. Аналіз даних. Дані аналізуються за допомогою алгоритмів машинного навчання. Алгоритми використовують попередньо підготовлені моделі для ідентифікації потенційних загроз на основі патернів та аномалій у даних. Результати аналізу передаються на етап виявлення загроз для подальшої оцінки та реагування.

6. Виявлення загроз. На цьому етапі система визначає, чи є знайдені патерни або аномалії в даних ознаками кібератак або інших загроз. Це може включати аналіз поведінкових аномалій, підозрілої активності або інших сигналів небезпеки. Якщо загроза виявлена, відповідні сигнали або алерти передаються на етап реагування. Також результати цього етапу можуть бути використані для оновлення моделі.

7. Реагування. Якщо виявлено загрозу, система ініціює дії для нейтралізації або мінімізації її впливу. Це може включати ізоляцію системи, блокування підозрілої активності або інші дії відповідно до налаштувань безпеки. Інформація про загрозу та результат реакції може передаватися на етап оновлення моделі для подальшого навчання системи.

8. Оновлення моделі. Система використовує нові дані, отримані в процесі виявлення загроз і реагування, для оновлення своїх моделей машинного навчання. Це дозволяє системі адаптуватися до нових загроз і покращувати точність виявлення в майбутньому. Оновлена модель знову використовується для аналізу даних, починаючи з етапу збору даних.

2.3 Алгоритми машинного навчання

1. Random Forest – це ансамблевий метод машинного навчання, який складається з великої кількості рішучих дерев, які працюють спільно. Кожне дерево у випадковому лісі генерується з випадкової вибірки даних, а остаточний результат визначається голосуванням усіх дерев. Цей метод є потужним інструментом для класифікації та регресії, і особливо ефективний у ситуаціях, коли дані мають велику кількість ознак.

Формула для обчислення критерію поділу (Gini(D)):

$$\boldsymbol{Gini(D) = 1 - \sum_{i=1}^{C} p_i^2}\ , \tag{2.7}$$

де $\boldsymbol{Gini(D)}$ – показник нечистоти вузла $\boldsymbol{D}$.

$\boldsymbol{C}$ – кількість класів.

$\boldsymbol{p_i}$ – частка об'єктів класу $\boldsymbol{i}$ у вибірці $\boldsymbol{D}$.

Основою для цієї моделі є концепція ентропії, яка використовується для вимірювання невизначеності в інформаційній теорії. Формула $\boldsymbol{Gini(D)}$ оцінює, наскільки добре вузол ділить вибірку на окремі класи, допомагаючи визначити, які ознаки найкраще розділяють дані.

Random Forest складається з великої кількості дерев рішень, кожне з яких будується на різних підмножинах даних. На кожному кроці побудови дерева обирається ознака, яка найкраще ділить вибірку, зменшуючи нечистоту вузла. При побудові дерева рішень, для кожного можливого поділу обчислюється $\boldsymbol{Gini}(\boldsymbol{D})$. Ознака, яка має найменшу $\boldsymbol{Gini}(\boldsymbol{D})$ (тобто, найменшу нечистоту), обирається для поділу вузла.

$\boldsymbol{Gini}(\boldsymbol{D})$ використовується для оцінки якості поділу даних у вузлі дерева. Менше значення $\boldsymbol{Gini}(\boldsymbol{D})$ вказує на те, що вузол містить більш однорідні класи. Обчислення $\boldsymbol{Gini}(\boldsymbol{D})$ є швидким і ефективним, що дозволяє швидко будувати дерева в Random Forest, навіть для великих наборів даних.

Завдяки цій формулі Random Forest можна застосовувати для виявлення аномалій у мережевих даних. Наприклад, дерева можуть навчатися на історичних даних, щоб визначати нормальну поведінку та виявляти відхилення від цієї норми, також вона допомагає в класифікації різних типів атак, аналізуючи мережеві пакети та журнали подій. Це дозволяє моделі краще розпізнавати різні типи кібератак і відрізняти їх від звичайної активності.

Random Forest є ансамблевим методом, який поєднує велику кількість окремих дерев рішень для підвищення точності та стійкості моделі. Цей алгоритм був обраний через свою здатність ефективно працювати з великим обсягом даних та високою кількістю ознак, що є типовим для задач кібербезпеки. Random Forest добре справляється з даними, що містять шум або пропуски, що дозволяє зменшити ймовірність помилкових спрацьовувань. Крім того, цей метод має високу інтерпретованість, що дозволяє легко пояснити рішення моделі та виявити ключові ознаки, які вплинули на результат.

2. Support Vector Machine – це алгоритм для класифікації даних, який знаходить гіперплощину, що найкраще відокремлює класи у просторі ознак. Вибір гіперплощини, яка максимізує відстань (маржу) між різними класами, допомагає SVM досягати високої точності класифікації.

Формула для функції втрат (L):

$$\boldsymbol{L(w, b, x_i, y_i) = max(0, 1 - y_i(w \times x_i + b)}\text{ ,} \qquad (2.8)$$

де $\boldsymbol{w}$ – вектор ваг.

$\boldsymbol{b}$ – зміщення.

$\boldsymbol{x_i}$ – вектор ознак.

$\boldsymbol{y_i}$ – мітка класу (+1 або −1).

Основою для цієї моделі є концепція максимізації маржі, яка використовується для досягнення оптимального розділення класів у просторі ознак. Функція втрат **L** мінімізує помилки класифікації, забезпечуючи оптимальне розділення класів.

Формула для функції втрат використовується для навчання SVM, щоб знайти гіперплощину, яка максимізує відстань між класами. Функція втрат допомагає мінімізувати помилки класифікації, забезпечуючи правильне розділення даних. Після оптимізації гіперплощини модель SVM може використовуватися для класифікації нових даних, забезпечуючи високу точність класифікації навіть у випадку, коли дані є нелінійно роздільними.

SVM з використанням даної формули може застосовуватися для виявлення аномалій у мережевих даних, це в свою чергу дозволяє моделі визначати відхилення від нормальної поведінки, що може свідчити про потенційну атаку. $\boldsymbol{L}$ допомагає в класифікації різних типів атак, аналізуючи мережеві пакети та журнали подій, що допоможе моделі краще розпізнавати різні типи кібератак і відрізняти їх від звичайної активності.

Support Vector Machine є потужним методом для класифікації даних, особливо коли мова йде про задачі з великою кількістю ознак і малою кількістю навчальних прикладів. SVM був обраний через свою здатність ефективно працювати в умовах високої розмірності простору ознак і забезпечувати хорошу

узагальнюючу здатність на нових даних. Цей метод також демонструє високу точність у виявленні аномалій, що є важливим аспектом у задачах кібербезпеки, де правильне розпізнавання загроз є критичним[23].

3. Neural Networks

Нейронні мережі – це тип алгоритму глибокого навчання, який імітує роботу людського мозку для обробки складних патернів і ознак у великих масивах даних. Нейронні мережі складаються з різних шарів нейронів, які трансформують вхідні дані, навчаються на них і приймають рішення на основі їх обробки[24].

Формула для активаційної функції (ReLU):

$$\boldsymbol{ReLU}(\boldsymbol{x}) = \boldsymbol{max}(\mathbf{0}, \boldsymbol{x}), \tag{2.9}$$

де $\boldsymbol{ReLU}(\boldsymbol{x})$ – вихід нейрона.

$\boldsymbol{x}$ – вхідне значення (сума зважених входів нейрона).

Основою для цієї моделі є концепція активаційних функцій, які визначають вихід нейронів у мережі. $\boldsymbol{ReLU}$ (Rectified Linear Unit) є однією з найпопулярніших активаційних функцій, яка забезпечує швидку та ефективну обробку даних у глибоких нейронних мережах.

$\boldsymbol{ReLU}$ додає нелінійність у модель, дозволяючи нейронній мережі вивчати складні патерни в даних, також вона є простою та швидкою у обчисленні, що робить її ефективною для тренування великих моделей на великих наборах даних, ще вона забезпечує розрідженість виходів (деякі нейрони можуть мати нульовий вихід), що робить модель більш стійкою до перенавчання.

Нейронні мережі з $\boldsymbol{ReLU}$ можуть використовуватися для виявлення аномалій у мережевих даних, визначаючи незвичайні патерни, які можуть свідчити про потенційні атаки, також $\boldsymbol{ReLU}$ допоможе у тренуванні глибоких нейронних мереж для класифікації різних типів атак, аналізуючи велику кількість ознак і виявляючи складні взаємозв'язки.

Нейронні мережі, зокрема глибокі нейронні мережі, мають здатність автоматично виділяти ознаки з сирих даних, що робить їх надзвичайно потужним

інструментом для аналізу складних і великомасштабних даних. Вибір нейронних мереж обумовлений їхньою здатністю навчатися на великих обсягах даних і виявляти складні, нелінійні закономірності, які можуть бути неочевидними для інших методів. Вони також можуть адаптуватися до нових загроз через процес додаткового навчання, що забезпечує їхню актуальність і ефективність у динамічному середовищі кіберзагроз[25].

Ці три методи були обрані через їхню взаємодоповнюваність і здатність вирішувати різні аспекти задачі виявлення та реагування на кіберзагрози. Random Forest забезпечує надійність і інтерпретованість, SVM дозволяє ефективно працювати з високорозмірними даними, а нейронні мережі забезпечують гнучкість та здатність до навчання на нових даних. Такий комплексний підхід дозволяє максимально ефективно використовувати переваги кожного з методів, мінімізуючи їхні недоліки, і забезпечує високу ефективність системи в цілому.

2.3.1 Навчання моделей.

Для створення ефективного аналітичного модуля для виявлення кіберзагроз вкрай важливим є процес навчання моделей. В основі цього процесу лежить використання історичних даних, які дозволяють моделям машинного навчання вивчити характерні шаблони та особливості загроз, з якими може стикатися система.

Для початку, необхідно зібрати значний обсяг історичних даних, що включають як нормальні, так і аномальні дії в системі. Ці дані можуть містити логи мережевої активності, телеметрію, журнал подій, тощо. Далі дані проходять процес попередньої обробки: очищення, нормалізація та форматування для забезпечення їх сумісності з обраними алгоритмами машинного навчання. Нормалізація даних здійснюється для приведення різних типів даних до єдиного стандарту, що полегшує їх обробку та аналіз. У багатьох випадках вхідні дані можуть бути незбалансованими, тобто певні класи даних представлені значно

більше, ніж інші. Для вирішення цієї проблеми можуть застосовуватися методи підвибірки або надвибірки, щоб забезпечити рівномірне представлення всіх класів.

Після підготовки даних їх необхідно розділити на дві частини: навчальну та тестову вибірки. Навчальна вибірка використовується для безпосереднього навчання моделі, а тестова – для оцінки її ефективності та точності після завершення навчання. Співвідношення між тренувальним і тестовим наборами зазвичай становить 80/20 або 70/30, але може варіюватися залежно від конкретних вимог і доступного обсягу даних.

Процес навчання моделі включає підбір оптимальних параметрів для алгоритмів машинного навчання. Для навчання використовуються попередньо обрані алгоритми машинного навчання, такі як Random Forest, SVM та нейронні мережі. Кожен алгоритм має свої параметри, які потребують налаштування для досягнення найкращих результатів. Модель проходить багаторазове навчання на тренувальних даних. Протягом цього процесу модель вчиться виявляти шаблони та аномалії, які можуть свідчити про наявність кібератак.

Після початкового тренування, моделі піддаються оптимізації шляхом налаштування гіперпараметрів. Це може включати вибір кількості дерев у Random Forest, регуляризацію у SVM або структуру нейронної мережі. Для оцінки узагальнюючої здатності моделі використовується метод перехресної валідації , що дозволяє перевірити модель на різних підмножинах даних. Однією з ключових цілей оптимізації є зниження ризику перенавчання , коли модель надмірно адаптується до тренувальних даних і втрачає здатність ефективно працювати на нових даних.

Процес тренування може бути автоматизованим, з використанням технік градієнтного спуску або інших методів оптимізації. Цей процес може бути досить тривалим, оскільки необхідно врахувати різні комбінації параметрів, щоб знайти ті, які забезпечують максимальну ефективність моделі. Для цього можуть використовуватися методи, такі як пошук по сітці або випадковий пошук. Під час тренування модель постійно оцінюється на основі валідаційного набору даних.

Це дозволяє виявити і виправити потенційні проблеми ще до завершення тренування.

Після завершення навчання модель перевіряється на тестових даних. Цей етап дозволяє оцінити, наскільки ефективно модель працює з новими даними, що не були використані під час навчання. Метрики, такі як точність, повнота, F1-міра, ROC-AUC використовуються для оцінки якості моделі. На основі оцінки обирається найкраща модель, яка буде використовуватися в системі для виявлення кібератак. У випадку потреби, можна також використовувати ансамблеві методи, поєднуючи кілька моделей для досягнення кращих результатів.

Для повноцінного розуміння оцінки ефективності та якості моделі, було розглянуто математичні формулу для обчислення градієнтів під час навчання моделей.

Формула для оновлення параметрів за допомогою градієнтного спуску:

$$\boldsymbol{\theta} := \boldsymbol{\theta} - \boldsymbol{\alpha}\boldsymbol{\nabla}_{\boldsymbol{\theta}}\boldsymbol{J}(\boldsymbol{\theta}), \tag{2.10}$$

де $\boldsymbol{\theta}$ – вектор параметрів моделі.

$\boldsymbol{\alpha}$ – швидкість навчання.

$\boldsymbol{\nabla}_{\boldsymbol{\theta}}\boldsymbol{J}(\boldsymbol{\theta})$ – градієнт функції втрат $\boldsymbol{J}(\boldsymbol{\theta})$ за параметрами $\boldsymbol{\theta}$.

Основою для цієї моделі є концепція градієнтного спуску, яка використовується для мінімізації функції втрат шляхом ітеративного оновлення параметрів у напрямку протилежному градієнту. Градієнтний спуск є одним з найпоширеніших алгоритмів оптимізації в машинному навчанні і використовується для навчання багатьох типів моделей, включаючи нейронні мережі, лінійні регресії та інші.

Градієнтний спуск використовується для мінімізації функції втрат $\boldsymbol{J}(\boldsymbol{\theta})$, яка оцінює, наскільки добре модель передбачає цільові значення. На кожній ітерації алгоритму градієнтного спуску параметри моделі $\boldsymbol{\theta}$ оновлюються за допомогою формули $\boldsymbol{\theta} := \boldsymbol{\theta} - \boldsymbol{\alpha}\boldsymbol{\nabla}_{\boldsymbol{\theta}}\boldsymbol{J}(\boldsymbol{\theta})$. Це означає, що параметри змінюються у напрямку, який зменшує значення функції втрат.

Швидкість навчання $\boldsymbol{\alpha}$ визначає розмір кроку, який робиться у напрямку градієнту. Занадто велика $\boldsymbol{\alpha}$ може призвести до пропуску мінімуму функції втрат, тоді як занадто мала $\boldsymbol{\alpha}$ може зробити процес навчання дуже повільним. Процес градієнтного спуску повторюється до тих пір, поки функція втрат не досягне мінімального значення або не перестане значно змінюватися. Це означає, що модель досягла оптимальних параметрів для даного набору даних.

Градієнтний спуск буде використовуватися для навчання моделей виявлення аномалій на основі історичних даних мережевої активності. Це дозволить моделі виявляти незвичайні патерни, які можуть вказувати на потенційні атаки. Навчання моделей класифікації атак за допомогою градієнтного спуску дозволить створити точні моделі, які можуть розпізнавати різні типи атак на основі мережевих даних. Ці моделі буде використано для прогнозування можливих загроз на основі історичних даних. Це дозволить системі кібербезпеки приймати проактивні заходи для запобігання атакам.

Оскільки характер кіберзагроз постійно змінюється, моделі необхідно регулярно оновлювати. Це передбачає постійне додавання нових даних до навчального набору та перевибір параметрів моделі для підтримки її актуальності та ефективності. Регулярні оновлення також дозволяють моделі адаптуватися до нових типів загроз, які могли з'явитися після початкового навчання. Такий підхід забезпечить ефективне навчання та адаптацію моделей машинного навчання, що дозволить системі постійно вдосконалювати можливості моделей у виявленні та реагуванні на кіберзагрози.

2.3.2 Оцінка результатів: Метрики ефективності алгоритмів, обробка результатів.

Ще 1 важливим етапом у розробці та впровадженні аналітичного модуля для виявлення кіберзагроз є оцінка результатів. Ефективність моделей

машинного навчання визначається на основі різних метрик, які дозволяють оцінити їхню здатність виявляти загрози, з мінімальною кількістю помилкових спрацювань та пропущених атак. Крім того, важливо забезпечити коректну обробку результатів, щоб отримані дані могли бути використані для подальших дій, таких як реакція на загрози або оновлення моделей.

Після проведення навчання модель, вони перевіряються на тестових даних, де обчислюються вищезазначені метрики. Оцінка результатів дозволяє визначити, наскільки модель добре працює у виявленні загроз у порівнянні з іншими підходами або моделями. Якщо результати не задовольняють вимоги, може бути проведено повторне налаштування параметрів або використані інші методи для покращення моделі.

Після оцінки метрик, результати обробляються для подальшого використання, до обробки результатів входить:

— Автоматичне створення звітів про виявлені загрози, що включають деталі про тип загрози, ймовірність її виникнення, та рекомендації щодо реакції.

— Створення графіків та діаграм, які показують роботу моделі, її ефективність та розподіл результатів.

— Використання результатів для коригування моделі та її параметрів у наступних етапах навчання.

Якщо результати показують, що модель потребує покращень, система може використовувати отримані результати для автоматичного оновлення параметрів моделі та повторного навчання. Крім того, результати можуть бути інтегровані з іншими модулями системи, наприклад, для автоматизації процесу реагування на загрози або для покращення збору даних.

Таким чином, дана частина роботи описує оцінку результатів, що є критично важливим процесом, який забезпечує ефективність та надійність аналітичного модуля, дозволяючи постійно підвищувати точність виявлення загроз і знижувати кількість помилкових спрацювань.

2.4 Інтерфейс та безпека системи

Цей розділ присвячений опису ключових аспектів, пов'язаних із взаємодією користувачів із системою та забезпеченням її безпеки. Ефективний інтерфейс є важливою складовою, що дозволить користувачу легко і швидко отримувати доступ до необхідної інформації та функцій. Особлива увага буде приділена забезпеченню зручності демонстрації функціоналу системи.

Безпека системи, зокрема захист даних і конфіденційності, є основним пріоритетом на всіх етапах розробки та експлуатації. У цьому розділі також розглядаються підходи до підтримки системи в актуальному стані, що включає регулярні оновлення програмного забезпечення та механізми контролю безпеки.

2.4.1 Взаємодія з користувачем

У початковій версії програмного рішення основна увага приділяється демонстрації ключових функціональних можливостей системи. Зважаючи на те, що система буде представлена на етапі розробки, інтерфейс буде розроблений таким чином, щоб забезпечити зручний доступ до основних функцій і наочно продемонструвати потенціал рішення.

Інтерфейс орієнтований на простоту та ефективність взаємодії з системою, дозволяючи оперативно запускати аналіз та отримувати результати. Це включає базові елементи управління, такі як кнопки для запуску процесів та виведення ключової інформації щодо виявлених загроз. Інтерфейс створений таким чином, щоб забезпечити легкість навігації та інтуїтивне розуміння процесу аналізу даних.

Що стосується засобів візуалізації результатів, вони будуть реалізовані у формі зрозумілих текстових повідомлень, що інформують про роботу алгоритмів та виявлені загрози. Водночас, передбачено можливість автоматичного створення графіків та діаграм, які дозволять наочно показати процес аналізу даних та

результати роботи системи, підкреслюючи її потенціал для подальшого розширення та інтеграції з більш комплексними аналітичними інструментами.

Таким чином, інтерфейс і візуалізація будуть побудовані з акцентом на демонстрації ключових можливостей системи, залишаючи простір для подальшого розвитку та вдосконалення функціоналу в майбутньому.

Формула для оцінки зручності інтерфейсу (Interface Usability Index, IUI):

$$\boldsymbol{IUI} = \frac{\sum_{\boldsymbol{i=1}}^{\boldsymbol{n}}(\boldsymbol{S_i} \times \boldsymbol{W_i})}{\boldsymbol{n}}, \tag{2.11}$$

де $\boldsymbol{IUI}$ – індекс зручності інтерфейсу.

$\boldsymbol{S_i}$ – оцінка зручності користувачем для завдання iii (від 1 до 5).

$\boldsymbol{W_i}$ – вага завдання і (від 0 до 1), яка відображає важливість цього завдання.

$\boldsymbol{n}$ – загальна кількість завдань, оцінених користувачами.

Основою для цієї моделі є концепція зваженого середнього, яка використовується для оцінки середнього рівня зручності інтерфейсу з урахуванням важливості кожного завдання.

Користувачі оцінюватимуть зручність використання інтерфейсу для різних завдань. Оцінки зручності (від 1 до 5) зважуються за важливістю завдань. Низькі значення $\boldsymbol{IUI}$ можуть вказувати на те, що певні завдання потребують покращення з точки зору зручності. Використання $\boldsymbol{IUI}$ дозволить виявляти і пріоритезувати зміни в інтерфейсі, щоб покращити загальний користувацький досвід.

2.4.2 Безпека та підтримка системи

Для забезпечення надійного функціонування системи та захисту оброблюваних даних, в нашому рішенні застосовуються кілька ключових заходів безпеки. Насамперед, використовується шифрування даних як під час їх зберігання, так і під час передачі. Це гарантує, що навіть у випадку

несанкціонованого доступу до системи, дані залишатимуться недоступними для зловмисників.

Контроль доступу до системи реалізується через багаторівневі механізми аутентифікації та авторизації, що дозволяють лише уповноваженим користувачам отримувати доступ до конфіденційної інформації та критичних функцій системи. Всі спроби доступу логуються, що дозволяє оперативно виявляти та реагувати на потенційні загрози.

Окрім цього, в системі впроваджено моніторинг і виявлення аномалій, що дозволяє відслідковувати підозрілу активність та забезпечує своєчасне виявлення й усунення загроз. Ці заходи дозволяють забезпечити високий рівень безпеки і конфіденційності даних, які обробляються в системі, а також надійно захистити саму систему від зовнішніх та внутрішніх загроз.

Також для забезпечення стабільного і ефективного функціонування системи буде підтримуватися процес оновлення та підтримки. Регулярні оновлення програмного забезпечення дозволяють інтегрувати нові функції, виправляти виявлені помилки, а також посилювати захист системи від нових кіберзагроз. Ці оновлення забезпечуються як для основних компонентів системи, так і для алгоритмів машинного навчання, які використовуються для аналізу та виявлення загроз.

Перевірка ефективності алгоритмів здійснюється на постійній основі, що дозволяє вчасно виявляти зниження їхньої продуктивності або точності. Це, у свою чергу, забезпечує можливість своєчасного оновлення моделей і параметрів, щоб підтримувати високу ефективність виявлення загроз.

Для підтримки системи безпеки в актуальному стані впроваджуються регулярні перевірки та аудит безпеки. Ці заходи допоможуть своєчасно виявити та усунути можливі вразливості, а також адаптувати систему до нових загроз і вимог.

Формула для оцінки рівня безпеки системи (System Security Level, SSL):

$$\boldsymbol{SSL = \frac{\sum_{i=1}^{m}(C_i \times V_i)}{m}}, \tag{2.12}$$

де $\boldsymbol{SSL}$ – рівень безпеки системи.

$\boldsymbol{C_i}$ – оцінка критичності вразливості iii (від 1 до 5).

$\boldsymbol{V_i}$ – оцінка вразливості iii (від 0 до 1), яка відображає ймовірність її експлуатації.

$\boldsymbol{m}$ – загальна кількість вразливостей, оцінених експертами з безпеки.

Основою для цієї моделі є концепція зваженого середнього, яка використовується для оцінки середнього рівня безпеки системи з урахуванням критичності та ймовірності експлуатації вразливостей.

Використання $\boldsymbol{SSL}$ дозволить зосередитися на найбільш критичних вразливостях та постійно покращувати рівень захищеності системи. Після оцінки критичності та ймовірності експлуатації різних вразливостей у системі. Вразливості з високими значеннями $\boldsymbol{C_i}$ та $\boldsymbol{V_i}$ повинні бути виправлені в першу чергу, щоб зменшити ризик для системи.

2.5 Висновки до розділу 2

У цьому розділі було детально розглянуто архітектуру програмного модуля для виявлення кібератак та основні теоретичні підходи, на яких базується його функціонування. Були проаналізовані компоненти системи, включаючи модулі збору даних, попередньої обробки, аналізу на основі алгоритмів штучного інтелекту та машинного навчання, а також модулі реагування та збереження результатів.

Розроблена архітектура модульної системи забезпечує гнучкість та масштабованість, що дозволяє ефективно інтегрувати її в існуючі IT-інфраструктури.

Застосування алгоритмів машинного навчання, таких як Random Forest, SVM та нейронні мережі, дозволяє досягати високої точності при виявленні аномалій у мережевому трафіку. Також розглянута побудова системи з

урахуванням зручності роботи для кінцевого користувача, що включає зрозумілий інтерфейс та можливість автоматичного реагування на загрози.

Загалом, архітектура програмного модуля спрямована на забезпечення надійного та ефективного захисту від кібератак, використовуючи сучасні методи аналізу даних та виявлення аномалій.

РОЗДІЛ 3
ПРОГРАМНА РЕАЛІЗАЦІЯ МОДУЛЯ ВИЯВЛЕННЯ КІБЕРАТАК В КОМП'ЮТЕРНИХ СИСТЕМАХ

3.1 Програмна реалізація

У цьому розділі буде представлено програмну реалізацію основних компонентів системи для виявлення кібератак. Розроблена система використовує сучасні методи штучного інтелекту та машинного навчання для аналізу мережевих даних у реальному часі та визначення аномалій. Основною метою є інтеграція декількох модулів, що відповідають за збір, обробку, аналіз даних та реагування на виявлені загрози.

3.1.1 Опис технологій, що використовуються

Python був обраний як основна мова програмування для реалізації системи виявлення кібератак через його популярність, універсальність та потужну екосистему інструментів для задач машинного навчання та обробки даних.

Python є однією з найпопулярніших мов програмування завдяки своїй простоті, читабельності та великій спільноті розробників, що робить його оптимальним вибором для вирішення складних завдань, пов'язаних із обробкою даних та реалізацією алгоритмів машинного навчання. У випадку розробки модуля для виявлення кібератак Python надає всі необхідні інструменти для швидкої та ефективної розробки. Його синтаксис інтуїтивно зрозумілий і дозволяє легко писати та підтримувати код.

Основні переваги Python для задач машинного навчання та обробки даних:

Широкий вибір бібліотек для машинного навчання:

scikit-learn – одна з найпоширеніших бібліотек для побудови моделей машинного навчання. Вона пропонує широкий вибір

алгоритмів (Random Forest, SVM, нейронні мережі тощо), що легко інтегруються в різні проекти.

TensorFlow та Keras – популярні бібліотеки для побудови та навчання нейронних мереж, що також підтримують навчання на великих даних.

joblib – бібліотека для збереження та завантаження натренованих моделей, що забезпечує зручність при використанні моделей у реальному часі.

Підтримка для обробки даних:

pandas – інструмент для роботи з табличними даними, що дозволяє легко завантажувати, обробляти та аналізувати великі обсяги інформації.

NumPy – основна бібліотека для роботи з багатовимірними масивами та виконання математичних операцій над даними.

Зручність інтеграції з системами моніторингу та збору даних:

psutil – бібліотека для збору даних про ресурси системи, що дозволяє ефективно реалізувати модуль збору даних для виявлення аномальної активності.

Гнучкість і можливість інтеграції з іншими мовами та технологіями:

Python легко інтегрується з C/C++, Java, а також системними командами, що забезпечує можливість побудови складних систем із використанням багатомодульної архітектури.

Інтуїтивний синтаксис та активна спільнота:

Простий та зрозумілий синтаксис Python дозволяє швидко розробляти та тестувати модулі. Це також робить його зручним для швидкого прототипування та внесення змін у код. Активна спільнота розробників забезпечує підтримку та оновлення бібліотек, а також вирішення виникаючих проблем.

Для реалізації системи виявлення кібератак важливу роль відіграють інструменти, що дозволяють збирати та обробляти дані з мережевих джерел у

режимі реального часу. У цьому проекті використовуються дві ключові бібліотеки: psutil для збору системних та мережевих даних, а також pandas для зручної роботи з табличними даними та їх подальшої обробки.

psutil – це бібліотека, що надає інтерфейси для збору інформації про системні ресурси та процеси. Вона дозволяє збирати дані про мережевий трафік, використання процесора, пам'яті та інших ключових ресурсів комп'ютера.

В проекті psutil дозволяє наступне:

— збирати інформацію про кількість відправлених та отриманих байтів, кількість відправлених та отриманих пакетів, а також інші мережеві метрики.

— за допомогою функцій psutil.net_io_counters() забезпечує збирання даних для подальшого аналізу в реальному часі, що допомагає виявляти кіберзагрози, такі як DDoS-атаки.

— використовуючи psutil.cpu_percent() та psutil.virtual_memory(), можна відслідковувати використання процесора та оперативної пам'яті.

— psutil дозволяє збирати дані як для одиночних процесів, так і для системи в цілому, що робить його зручним для масштабованих рішень, таких як системи моніторингу безпеки.

pandas – це потужна бібліотека для роботи з табличними даними, яка широко використовується в задачах аналізу даних та обробки великих обсягів інформації. Вона є ключовим компонентом для зберігання та попередньої обробки зібраних мережевих даних перед їх аналізом за допомогою алгоритмів машинного навчання.

В проекті pandas дозволяє наступне:

— Завдяки pandas.read_csv() дозволяє з легкістю завантажувати дані із зовнішніх файлів у форматі CSV, що є стандартним для багатьох інструментів моніторингу та збору даних.

— Зібрані мережеві дані можуть бути збережені в об'єкті DataFrame, що надає широкі можливості для фільтрації, сортування та попередньої обробки.

— Функції pandas дозволяють виконувати такі операції, як очищення даних від пропущених значень, заміна відсутніх значень, нормалізація та форматування даних.

— pandas дозволяє обробляти великі обсяги даних (наприклад, мережеві журнали), використовуючи DataFrame, який оптимізовано для швидкого доступу та модифікації даних. Це робить її ідеальною для завдань, де потрібно обробляти тисячі або навіть мільйони записів мережевого трафіку.

— pandas дозволяє зберігати та завантажувати дані з різних форматів, таких як CSV, Excel та SQL-бази даних. Це забезпечує легку інтеграцію з іншими модулями системи та сторонніми інструментами моніторингу.

Для виявлення кібератак на основі аналізу мережевого трафіку використовується машинне навчання, яке дозволяє системі автоматично навчатися на основі історичних даних і робити прогнози щодо майбутніх подій. Однією з основних бібліотек, яка використовується для реалізації алгоритмів машинного навчання в цьому проекті, є scikit-learn.

scikit-learn – це одна з найпопулярніших бібліотек Python для реалізації алгоритмів машинного навчання. Вона надає інструменти для класифікації, регресії, кластеризації, зниження вимірностей, підбору параметрів і оцінки моделей. Бібліотека широко використовується в задачах аналізу даних, обробки сигналів, фінансових прогнозів, біомедицини та багатьох інших галузях.

scikit-learn був вибраний для проекту оскільки у бібліотека надає доступ до багатьох стандартних алгоритмів машинного навчання, таких як:

1. Random Forest – використовується для класифікації та виявлення аномалій у мережевих даних. Цей метод працює на основі ансамблю рішень дерев, що дозволяє знизити ризик помилок через перенавчання.
2. Support Vector Machine – застосовується для класифікації високорозмірних даних. SVM добре справляється з проблемами, пов'язаними з нелінійними даними, що часто трапляються при аналізі мережевого трафіку.

3. MLPClassifier (нейронні мережі) – використовується для задач багатокласової класифікації. Це дозволяє системі виявляти складні патерни в даних, такі як атаки нульового дня або приховані загрози.

scikit-learn має простий інтерфейс для налаштування, навчання та використання моделей машинного навчання. Зокрема, для навчання моделі потрібно лише кілька рядків коду:

```
model = RandomForestClassifier(n_estimators=100)
model.fit(X_train, y_train)
```

Легко інтегрується з іншими бібліотеками, такими як pandas, що дозволяє зручно підготовляти та передавати дані для навчання моделей.

scikit-learn надає набір інструментів для оцінки якості навченої моделі, для цього використовуються наступні метрики метрики:

Точність (accuracy) – визначає кількість правильних передбачень моделі.

F1-score – показник гармонійного середнього між точністю та повнотою, що допомагає оцінити якість класифікації в умовах незбалансованих даних.

Precision (Точність) – це частка правильних позитивних передбачень серед усіх передбачених позитивних класів. Вона показує, наскільки модель точна у визначенні позитивних випадків

Recall – це частка правильно передбачених позитивних класів серед усіх фактичних позитивних класів. Повнота показує, як добре модель знаходить всі реальні позитивні приклади:

ROC-AUC – використовується для оцінки якості класифікації на основі порівняння чутливості та специфічності.

GridSearchCV та RandomizedSearchCV – інструменти, що дозволяють виконувати автоматизований підбір гіперпараметрів для моделей машинного навчання. Це дає змогу вибрати найкращі параметри для

покращення результатів класифікації.

```
from sklearn.model_selection import GridSearchCV
param_grid = {'n_estimators': [50, 100, 150]}
grid_search = GridSearchCV(RandomForestClassifier(), param_grid, cv=5)
grid_search.fit(X_train, y_train)
```

PCA (Principal Component Analysis) – метод зниження розмірності даних, що використовується для зменшення кількості ознак мережевого трафіку без втрати важливої інформації. Це покращує продуктивність моделей і робить їх більш стійкими до перенавчання[26].

scikit-learn ефективно працює з великими наборами даних і забезпечує паралельну обробку, що дозволяє масштабувати рішення для використання у великих мережах із великим обсягом трафіку. Таким чином, завдяки використанню scikit-learn система виявлення кібератак здатна швидко навчатися на основі великих наборів даних, аналізувати мережевий трафік у реальному часі та надійно виявляти потенційні загрози на основі алгоритмів машинного навчання.

Ефективна система виявлення кібератак повинна не тільки виявляти загрози, але й створювати докладні логи та звіти для подальшого аналізу подій. Для цього використовуються такі інструменти, як logging для створення логів подій програми та SQLAlchemy для збереження логів у базі даних SQLite.

logging – це стандартний модуль Python, який забезпечує гнучкі можливості для ведення логів. Він дозволяє фіксувати всі важливі події в роботі програми, що робить систему прозорою та зручною для моніторингу. Перевагами використання logging є наступне:

Логи можуть бути різних рівнів: від DEBUG для детального відстеження до ERROR для фіксації помилок. Це дозволяє налаштовувати логування відповідно до потреб.

Логи можуть бути записані у файли, відправлені по мережі або виведені на консоль, що робить цей модуль універсальним для будь-яких потреб.

В системі логування використовується для фіксації збору даних, виявлення аномалій, сповіщення адміністратора про загрози та запису інших подій.

```
import logging
logging.basicConfig(filename='system.log',
level=logging.INFO, format='%(asctime)s - %(levelname)s - %(message)s')
```

Логи допомагають фіксувати всі ключові події, як-то успішний збір даних, виявлення аномалій, відправку повідомлень про загрози. Це дозволяє оперативно реагувати на потенційні загрози та мати повний контроль над системою.

Наприклад, при виявленні аномалії в мережевому трафіку записується відповідне повідомлення в лог-файл.

```
logging.info("Аномалія виявлена і IP заблоковано.")
```

logging легко інтегрується з іншими модулями та бібліотеками, такими як SQLAlchemy. Це дозволяє передавати логи не тільки в локальні файли, але й у базу даних для довготривалого зберігання та подальшого аналізу.

SQLAlchemy – це популярна ORM-бібліотека для Python, яка дозволяє працювати з базами даних через об'єктно-орієнтований інтерфейс. У цьому проекті SQLAlchemy використовується для збереження логів у базі даних SQLite, що дозволяє зберігати логи у структурованому вигляді.

Перевагами використання SQLite для логування є:

— SQLite – це легка база даних, яка не потребує окремого сервера і працює безпосередньо з файлами на локальному диску. Це дозволяє легко інтегрувати її в програму без складної конфігурації.

— Всі логи системи (події збору даних, виявлення аномалій,

сповіщення) зберігаються в таблицях бази даних, що робить їх доступними для подальшого аналізу:

```
from sqlalchemy import create_engine, Column, Integer, String, DateTime
from sqlalchemy.orm import sessionmaker, declarative_base
import datetime
Base = declarative_base()
class LogEntry(Base):
    __tablename__ = 'logs'
    id = Column(Integer, primary_key=True)
    event = Column(String)
    event_type = Column(String)
    timestamp = Column(DateTime)
```

Кожна важлива подія, як-то успішний збір даних або виявлення загрози, записується в базу даних за допомогою SQLAlchemy:

```
def save_log(event, event_type):
    log_entry = LogEntry(event=event, event_type=event_type,
    timestamp=datetime.datetime.now())
    session.add(log_entry)
    session.commit()
```

Це дозволяє зберігати логи у структурованому вигляді та проводити їх аналіз через запити до бази даних. Наприклад, можна фільтрувати логи за датою, типом події або важливістю, що полегшує подальший аналіз системи.

Використання SQLAlchemy дозволяє легко масштабувати систему та зберігати логи не лише у локальних базах, але й у віддалених або хмарних базах даних. Це дозволяє забезпечити надійність зберігання даних навіть при великих обсягах подій.

На відміну від лог-файлів, що можуть бути видалені або переписані, логи у базі даних можуть зберігатися тривалий час, що дозволяє проводити історичний аналіз та оцінювати ефективність системи безпеки протягом часу.

У системах виявлення кібератак важливо забезпечити оперативне інформування адміністратора про підозрілу активність у мережі або на сервері. Для цього використовуються автоматичні повідомлення електронною поштою, які надсилаються за допомогою бібліотеки smtplib. Для безпечного зберігання конфіденційних даних, таких як email-кредeнціали, використовується бібліотека dotenv.

При виявленні підозрілої або аномальної активності в системі, система автоматично відправляє повідомлення адміністратору за допомогою smtplib. Це дозволяє миттєво інформувати відповідальну особу про загрозу, на яку необхідно негайно відреагувати:

```
import smtplib
from email.mime.text import MIMEText
def send_email_alert(message, subject="Security Alert"):
    msg = MIMEText(message)
    msg['Subject'] = subject
    msg['From'] = EMAIL_USER
    msg['To'] = 'admin@example.com'
    with smtplib.SMTP('smtp.gmail.com', 587) as server:
        server.starttls()  # Захищене з'єднання
        server.login(EMAIL_USER, EMAIL_PASSWORD)
        server.send_message(msg)
```

Використовується захищене з'єднання за допомогою starttls(), що забезпечує безпечну передачу даних по каналу зв'язку.

Повідомлення містить інформацію про виявлену загрозу, а також рекомендації щодо подальших дій. Це дозволяє адміністратору своєчасно вжити необхідних заходів.

Використання smtplib дозволяє повністю автоматизувати процес сповіщення, що виключає можливість втрати важливої інформації або затримки у реагуванні на загрозу.

python-dotenv – це бібліотека, яка допомагає завантажувати змінні середовища з файлів .env. Це дозволяє зберігати конфіденційну інформацію окремо від основного коду, що забезпечує додатковий рівень безпеки.

Файли .env використовуються для зберігання чутливих даних, таких як логіни, паролі або API ключі. Завдяки dotenv, ці дані не потрібно жорстко кодувати у програмі, що зменшує ризик їх випадкового витоку:

EMAIL_USER=your_email@example.com
EMAIL_PASSWORD=your_password
В програмі ці дані завантажуються через **dotenv:**

```
from dotenv import load_dotenv
import os
load_dotenv()  # Завантаження змінних середовища з файлу .env
EMAIL_USER = os.getenv('EMAIL_USER')
EMAIL_PASSWORD = os.getenv('EMAIL_PASSWORD')
```

Файли .env зазвичай не включаються до репозиторіїв або загальнодоступних місць, що забезпечує додаткову безпеку. Це робить такі файли недоступними для сторонніх осіб, знижуючи ризик компрометації облікових даних.

dotenv забезпечує безпечний механізм для роботи з обліковими даними та іншою чутливою інформацією, що важливо для безпечного функціонування системи.

Використання dotenv дозволяє легко змінювати налаштування без необхідності змінювати сам код програми. Наприклад, можна оновити email-креденціали або інші конфіденційні дані без зміни вихідного коду.

Для підвищення зручності використання системи виявлення кібератак було вирішено створити графічний інтерфейс за допомогою бібліотеки Tkinter. Це дозволяє користувачам працювати з програмою без необхідності використовувати командний рядок, що значно спрощує процес вибору джерел даних та перегляду результатів аналізу.

Tkinter – це стандартна бібліотека Python для створення графічних інтерфейсів. Вона надає інструменти для розробки вікон, кнопок, текстових полів та інших елементів інтерфейсу, що робить її ідеальним рішенням для розробки простих і функціональних програм з графічним інтерфейсом.

Завдяки Tkinter, користувач може легко вибрати джерело даних для аналізу – це можуть бути як зібрані в реальному часі дані, так і дані з CSV-файлів:

self.real_time_button = Button(root, text="Збір даних у реальному часі", command=self.collect_real_time_data, width=40, height=2)

self.csv_button = Button(root, text="Завантажити дані з CSV файлу", command=self.load_csv_file, width=40, height=2)

Інтерфейс забезпечує інтуїтивно зрозумілий підхід, де користувач має змогу натиснути відповідну кнопку для вибору дії. Це значно спрощує роботу з програмою та усуває необхідність вводити команди вручну.

Після вибору джерела даних та завершення процесу аналізу, результати виводяться у текстове поле інтерфейсу. Для цього використовується елемент Text, що дозволяє відображати великі обсяги тексту з можливістю прокрутки:

self.report_text = Text(root, height=20, width=90, yscrollcommand=self.scrollbar.set)

self.report_text.insert(END, "Результати аналізу...")

Це рішення забезпечує користувача доступом до всієї необхідної інформації в одному вікні, без потреби відкривати сторонні програми або файли.

Користувач може взаємодіяти з програмою в реальному часі, отримуючи повідомлення про стан системи, аналіз даних та можливі загрози. Інтерфейс також підтримує виведення різних типів повідомлень, наприклад інформаційних або попереджень про виявлення аномалій.

Для прикладу, після виявлення аномалії в системі відображається повідомлення:

self.log_to_gui("[ALERT] Аномалія виявлена! IP заблоковано.\n")

Така функціональність дозволяє своєчасно інформувати користувача про загрози та дії, що були виконані для їх нейтралізації.

Tkinter надає можливість легко розширювати інтерфейс, додаючи нові функції, такі як додаткові джерела даних або нові параметри аналізу. Це забезпечує гнучкість у розробці та можливість адаптації системи до нових вимог.

Інтерфейс також може бути адаптований для виведення більш детальних звітів або інтеграції з іншими системами для моніторингу та аналізу.

Використання Tkinter не вимагає додаткових бібліотек або модулів для роботи на більшості операційних систем, оскільки вона є частиною стандартної бібліотеки Python. Це знижує системні вимоги до додатку та забезпечує сумісність з різними платформами. Таким чином, використання Tkinter у проекті дозволяє створити зручний і функціональний графічний інтерфейс, що забезпечує інтуїтивне керування системою виявлення кібератак та доступ до результатів її роботи.

3.1.2 Опис процесу реалізації програмного модуля.

Основна архітектура модуля складається з кількох компонентів, які забезпечують збір, обробку, аналіз даних за допомогою методів машинного навчання, а також реагування на виявлені загрози. Кожен модуль виконує свою специфічну функцію, інтегруючись у загальну систему для забезпечення безперервного моніторингу та захисту комп'ютерної мережі.

1. Модуль збору даних

Модуль збору даних виконує ключову функцію виявлення загроз, отримуючи інформацію про мережеву активність у реальному часі або завантажуючи її з попередньо збережених файлів CSV. Основні сценарії роботи модуля передбачають два режими:

Збір даних у реальному часі:

Модуль використовує бібліотеку psutil для моніторингу мережевої активності. Psutil забезпечує збір метрик, таких як кількість відправлених і отриманих байтів та пакетів. Цей процес відбувається періодично, наприклад, кожні 5 секунд, і результат зберігається в структуру даних pandas DataFrame для подальшої обробки та аналізу.

Приклад зібраних даних включає:

- Кількість відправлених байтів
- Кількість отриманих байтів
- Кількість відправлених і отриманих пакетів
- Тривалість потоку даних
- Дата і час кожного запису

Завантаження даних із CSV:

Модуль також підтримує можливість завантаження даних із файлів у форматі CSV, що дозволяє використовувати попередньо збережені набори даних. Для цього використовується бібліотека pandas, яка дозволяє легко зчитувати CSV-файли та перетворювати їх у зручну для обробки структуру даних DataFrame.

В даному модулі були інтегровані бібліотеки**:**

— **psutil** відповідає за збір мережевих метрик у реальному часі.

— **pandas** використовується для збереження, обробки та завантаження даних із CSV-файлів, що дає змогу аналізувати мережеву активність як у реальному часі, так і з архівних даних.

Цей модуль є основою для подальших етапів обробки, таких як попередня обробка даних і аналіз на основі алгоритмів машинного навчання.

2. Модуль попередньої обробки даних

Модуль попередньої обробки даних виконує ключові операції з очищення, нормалізації та зменшення розмірності даних перед їх передачею до алгоритмів машинного навчання. Його основне завдання полягає в тому, щоб підготувати зібрані або завантажені дані для подальшого аналізу, забезпечуючи, щоб вони були коректними та придатними для використання в моделях.

Очищення даних:

— Видалення непотрібних або зайвих стовпців (наприклад, стовпці з часовими мітками).

— Обробка пропущених значень шляхом їх заміни на відповідні значення або видалення рядків із пропущеними даними.

— Видалення некоректних значень, таких як нескінченності (inf) або неопрацьовані NaN (Not a Number).

Для нормалізації даних використовується метод StandardScaler із бібліотеки scikit-learn для нормалізації ознак. Цей метод стандарталізує кожну ознаку, перетворюючи її на значення зі середнім рівнем 0 і стандартним відхиленням 1. Це необхідно для покращення роботи моделей машинного навчання, особливо для алгоритмів, що залежать від масштабування даних, таких як SVM та нейронні мережі.

Для ефективної обробки великих обсягів даних та запобігання перенавчанню моделей, використовується метод PCA (Principal Component Analysis) з scikit-learn.

Завдяки PCA можна Зменшити кількість ознак у даних, вибираючи найбільш важливі компоненти, які містять найбільшу частину інформації. Це не тільки знижує складність моделей, але й пришвидшує їх навчання, зберігаючи при цьому важливі структури даних. Зменшення розмірності виконується з використанням до 20 основних компонентів (якщо кількість ознак дозволяє). Це значно покращує продуктивність і точність моделей.

Основними функціями цього модуля є:

— Очищення даних (видалення або заміна некоректних значень).

— Нормалізація ознак за допомогою StandardScaler.

— Зменшення розмірності з використанням PCA для спрощення моделі та збереження ключової інформації.

Цей модуль забезпечує готовність даних до подальшого етапу аналізу на основі алгоритмів машинного навчання, дозволяючи моделям працювати коректно та ефективно.

3. Модуль аналізу за допомогою ШІ

Модуль аналізу за допомогою штучного інтелекту є основною частиною системи, що забезпечує виявлення аномальної активності або загроз на основі зібраних та попередньо оброблених даних. Цей модуль використовує кілька алгоритмів машинного навчання для класифікації даних і виявлення потенційних кіберзагроз. Під час роботи системи використовуються вже попередньо навчені моделі, які були створені на основі історичних даних із мітками. Модуль дозволяє як тренувати нові моделі, так і завантажувати вже натреновані моделі з файлів для аналізу нових даних у реальному часі.

Модуль підтримує три основні типи моделей:

RandomForest:

— Алгоритм створює ансамбль дерев рішень, кожне з яких приймає незалежне рішення про клас зразка.

— В кінці модуль об'єднує рішення всіх дерев і приймає кінцевий прогноз на основі більшості голосів.

— RandomForest є стійким до перенавчання, що робить його

надійним для задач класифікації в кібербезпеці.

SVM:

— Алгоритм побудови гіперплощини, яка максимально розділяє зразки двох класів.

— SVM добре працює з невеликими та високорозмірними наборами даних і забезпечує високу точність у класифікації кіберзагроз.

MLP:

— Це нейронна мережа, що має кілька шарів між вхідними та вихідними даними.

— MLP ефективна для нелінійних задач і може адаптуватися до складних структур даних, роблячи її корисною для виявлення складних кіберзагроз.

Оцінка моделей виконується за допомогою кількох метрик, що дозволяє зрозуміти, наскільки ефективно система виявляє загрози. Основними функціями цього етапу є:

— Тренування моделей, яке виконується на історичних даних з мітками.

— Під час аналізу в реальному часі завантажуються попередньо натреновані моделі.

— Оцінка ефективності, за допомогою використання метрик для аналізу результатів класифікації та точності роботи моделей.

— Аналіз в реальному часі за допомогою застосування моделей для виявлення аномалій у даних зібраних у реальному часі.

— Для оцінки ефективності моделей машинного навчання використовуються різні метрики, такі як точність, повнота, точність передбачення, F1-міра, та площа під кривою ROC. Ці метрики дозволяють об'єктивно оцінити якість роботи моделі при виявленні аномалій або кіберзагроз.

Далі представлено математичні формули для кожної з метрик і приклад їх розрахунку на тестових даних

Точність (Accuracy)

Відношення правильних передбачень до загальної кількості передбачень. Ця метрика дозволяє оцінити загальну ефективність моделі.

$$\boldsymbol{Accuracy} = \frac{\boldsymbol{TP} + \boldsymbol{TN}}{\boldsymbol{TP} + \boldsymbol{TN} + \boldsymbol{FP} + \boldsymbol{FN}}, \quad (3.1)$$

де $\boldsymbol{TP}$ – кількість істинно позитивних результатів (правильно класифіковані атаки).

$\boldsymbol{TN}$ – кількість істинно негативних результатів (правильно класифіковані не атаки).

$\boldsymbol{FP}$ – кількість хибнопозитивних результатів (не атаки, які помилково класифіковані як атаки).

$\boldsymbol{FN}$ – кількість хибнонегативних результатів (атаки, які помилково класифіковані як не атаки).

Повнота (Recall)

Частка справжніх позитивних випадків, які були правильно виявлені моделлю. Важлива метрика для оцінки здатності моделі виявляти всі можливі загрози.

$$\mathbf{Recall} = \frac{\mathbf{TP}}{\mathbf{TP} + \mathbf{FN}} \quad (3.2)$$

Точність позитивних передбачень (Precision)

Частка правильних позитивних передбачень серед усіх випадків, які модель визнала загрозами. Вона дозволяє оцінити кількість помилкових спрацювань.

$$\mathbf{Precision} = \frac{\mathbf{TP}}{\mathbf{TP} + \mathbf{FP}} \quad (3.3)$$

F1-міра (F1-score)

Гармонійне середнє між точністю позитивних передбачень та повнотою. Це інтегрована метрика, яка враховує як хибнопозитивні, так і хибнонегативні результати.

$$\mathbf{F1} = \mathbf{2} \times \frac{\mathbf{Precision} \times \mathbf{Recall}}{\mathbf{Precision} + \mathbf{Recall}} \quad (3.4)$$

Площа під кривою ROC (AUC-ROC)

Площа під кривою (AUC-ROC) є мірою здатності моделі розрізняти позитивні та негативні класи. Значення варіюються від 0.5 (випадкове передбачення) до 1 (ідеальне передбачення).

Для прикладу припустимо, після аналізу тестових даних ми отримали наступну матрицю плутанини:

Таблиця 3.1

Приклад даних з матриці плутунини

	Аномалія (передбачено)	Норма (передбачено)
Аномалія (реальна)	30	10
Норма (реальна)	5	55

$$\boldsymbol{TP = 30, FN = 10, FP = 5, TN = 55}$$

1. $\text{Accuracy} = \frac{30+55}{30+55+5+10} = 0.85$

2. $\text{Recall} = \frac{30}{30+10} = 0.75$

3. $\text{Precision} = \frac{30}{30+5} = 0.857$

4. $\text{F1} = 2 \times \frac{0.857 \times 0.75}{0.857+0.75} \approx 0.799$

5. Якщо ми побудуємо графік ROC-кривої, значення AUC-ROC для даної моделі буде близько 0.89, що свідчить про хорошу здатність моделі розрізняти нормальні та аномальні дані.

Також розібрав математики для моделей штучного навчання на прикладі формул які розглядалися в 2 розділі.

1. Random Forest

Формула для обчислення критерію поділу, див формулу 2.7:

Приклад розрахунку:

Припустимо, що у вузлі $\boldsymbol{D}$ є дві категорії – аномальні та нормальні дані. Нехай 70% – це нормальні дані, а 30% – аномальні.

Тоді:

$p_i = 0{,}7$ (нормальні), $p_2 = 0{,}3$ (нормальні)

Підставимо значення у формулу:

$$Gini(D) = 1 - (0.7^2 - 0.3^2) = 0.42$$

Отже, Gini-критерій для цього вузла дорівнює 0.42. Чим менший цей показник, тим чистіший вузол (тобто менше змішання класів).

2. Support Vector Machine

Формула для функції втрат, див. формулу 2.8:

Припустимо, що маємо наступні значення:

w = [0.5,−0.3]

b = [2,1]

x_i = 0.1

y_i = 1 (позитивна класова мітка)

Обчислюємо:

$w \times x_i$ = *(0.5×2)+(−0.3×1)= 1−0.3= 0.7*

$w \times x_i + b$ = *0.7+0.1= 0.8*

$$L(w, b, x_i, y_i) = max\big(0{,}1 - (1 \times 0.8)\big) = \max(0, 0.2) = 0.2$$

Отже, функція втрат дорівнює 0.2, що свідчить про те, що класифікатор близький до коректної класифікації. Мета алгоритму – мінімізувати цю функцію втрат, щоб розмежувати класи між собою з максимальною маржею.

3. Нейронні мережі

Формула для активаційної функції, див. формулу 2.9:

Приклад розрахунку:

Нехай вхідне значення для нейрона становить $\boldsymbol{x}$ = −2.5. Використаємо активаційну функцію ReLU:

$$ReLU(-2.5) = max(0, -2.5) = 0$$

Якщо значення $\boldsymbol{x}$ = 3.4, то

$$ReLU(3.4) = max(0,3.4) = 3.4$$

Якщо вхідне значення більше 0, вихід нейрона дорівнює значенню $\boldsymbol{x}$, тобто 3.4.

ReLU дозволяє пропускати лише додатні значення, що допомагає уникати проблеми затухаючих градієнтів, характерної для інших активаційних функцій.

У кожній моделі використовується своя математична логіка. Random Forest будує рішення на основі критерію Gini для визначення найбільш інформативного поділу. SVM використовує функцію втрат для максимізації маржі між класами. Нейронні мережі використовують функцію активації ReLU для нелінійного перетворення вхідних даних.

4. Модуль реагування

Модуль реагування є критичною частиною системи, яка дозволяє автоматично реагувати на виявлені аномалії та кіберзагрози. Його функціонал забезпечує оперативне надсилання сповіщень адміністраторам і виконання заходів для мінімізації можливих загроз, таких як блокування підозрілих IP-адрес.

Як тільки система виявляє аномалії або потенційну кіберзагрозу, необхідно негайно сповістити адміністратора або відповідальну особу. Для цього використовується бібліотека smtplib, яка дозволяє надсилати електронні листи через SMTP-сервери.

Алгоритм надсилання email сповіщень:

1. Після виявлення аномалії в модулі аналізу система формує текстове повідомлення, яке містить опис загрози або підозрілої активності.
2. Для захисту облікових даних використовується бібліотека dotenv, яка дозволяє зберігати конфіденційну інформацію у файлі .env.
3. Система встановлює з'єднання з SMTP-сервером і аутентифікується, використовуючи збережені email-кредeнціали.

4. Після успішної аутентифікації система надсилає електронний лист із заголовком та вмістом до адміністратора.

Приклад сповіщення:

send_email_alert("Аномальна активність виявлена! Перевірте систему.", "Security Alert")

Основними перевагами такої системи є те, що сповіщення надходять на електронну пошту в реальному часі, що дозволяє оперативно реагувати на потенційні загрози, а також завдяки використанню .env файлів для зберігання облікових даних, підвищується безпека процесу надсилання сповіщень.

Одним із важливих заходів у відповідь на виявлення кіберзагроз є блокування підозрілих IP-адрес. Для цього використовується інструмент netsh або відповідні команди на інших операційних системах, що дозволяють динамічно блокувати IP-адреси на рівні брандмауера.

Алгоритм блокування IP-адреси:

1. Після виявлення аномальної активності система аналізує мережевий трафік та визначає IP-адресу, з якої може надходити загроза.
2. Викликається команда netsh advfirewall firewall add rule, яка блокує вхідний трафік із підозрілої IP-адреси на рівні системного брандмауера.
3. Система записує подію про блокування IP-адреси в лог-файл.

Приклад блокування IP:

block_ip("192.168.1.100")

Модуль реагування працює наступним чином:

1. Модуль аналізу на основі ШІ визначає аномалію у вхідних даних.
2. Як тільки загроза ідентифікована, модуль реагування негайно надсилає повідомлення на електронну пошту адміністратора.
3. Система блокує підозрілий IP, щоб припинити можливі атаки

на систему.

4. Всі дії (надсилання сповіщень, блокування IP) записуються в лог для подальшого аналізу.

5. Модуль зберігання даних

Однією з ключових вимог до системи є ведення журналу подій та збереження даних про виявлені аномалії й дії, що були вжиті для їхньої нейтралізації. Для цього використовується SQLAlchemy – бібліотека для роботи з базами даних у Python.

Кожен запис лог-файлу містить:

— Тип події: Наприклад, "Security Alert", "Data Collection".

— Час події: Точна дата та час, коли подія відбулася.

— Опис події: Детальний опис дії, наприклад, "IP-адреса 192.168.1.100 заблокована".

Для збереження логів у базі даних використовується SQLAlchemy із базою даних SQLite. Кожна подія записується в базу за допомогою створеної моделі LogEntry. Приклад коду для збереження події:

```
def save_log(event, event_type, timestamp):
    log_entry = LogEntry(event=event, event_type=event_type, timestamp=timestamp)
    session.add(log_entry)
    session.commit()
    logging.info(f"Лог збережено: {event} (Тип: {event_type})")
```

Для забезпечення повного контролю за роботою системи передбачена функція автоматичного створення звітів. Звіти дозволяють адміністраторам отримувати детальну інформацію про всі загрози та дії, які були вжиті для їхнього усунення.

Алгоритм створення звіту є наступним:

1. Після виявлення загроз модуль аналізу на основі ШІ формує

результати роботи системи – які загрози були виявлені та з якими характеристиками.

2. Модуль реагування надає інформацію про те, які дії були виконані системою (наприклад, блокування IP або надсилання сповіщень).

3. Система надає рекомендації адміністраторам щодо можливих подальших дій (наприклад, додаткові перевірки системи).

Звіти автоматично генерується у вигляді текстового файлу, який зберігається в спеціальній папці для звітів. Наприклад:

```
def create_report(analysis_results, actions_taken, recommendations):
    report_file = f'reports/report_{datetime.now().strftime("%Y-%m-%d_%H-%M-%S")}/security_report.txt'
    with open(report_file, 'w') as file:
        file.write("Звіт про безпеку системи\n")
        file.write(f"Дата: {datetime.now().strftime('%Y-%m-%d %H:%M:%S')}\n\n")
        file.write("Результати аналізу:\n")
        for result in analysis_results:
            file.write(f"- {result}\n")

        file.write("\nВжиті дії:\n")
        for action in actions_taken:
            file.write(f"- {action}\n")

        file.write("\nРекомендації:\n")
        for recommendation in recommendations:
            file.write(f"- {recommendation}\n")

    logging.info(f"Звіт створено у файлі: {report_file}")
    return report_file
```

Загальний процес даного модуля можна описати наступним чином:

1. Кожна важлива подія системи записується в базу даних для подальшого аналізу.
2. Після завершення кожного сеансу аналізу загроз система генерує текстовий звіт із описом виявлених загроз та вжитих заходів.
3. Усі дії та звіти можуть бути доступні через логи або відправлені адміністраторам через електронну пошту для швидкого реагування.

6. Головний код

Головний код є центральним елементом програми, який відповідає за управління всіма модулями системи та взаємодію між ними. Він забезпечує послідовний запуск основних процесів, таких як збір даних, їх обробка, аналіз на основі моделей машинного навчання, а також реагування на виявлені загрози. Головний код також реалізує графічний інтерфейс, який полегшує користувачу доступ до функцій системи.

Основними функціями головного коду є:

— Після запуску програми користувачу пропонується вибрати джерело даних для аналізу. Це може бути як реальний трафік, що збирається в режимі реального часу, так і дані з файлу CSV.

— За допомогою бібліотеки psutil збираються показники мережевої активності, після чого ці дані обробляються та передаються на аналіз.

— Дані з файлу завантажуються за допомогою бібліотеки pandas, і вони також передаються на попередню обробку перед аналізом.

— Після отримання даних, викликається модуль попередньої обробки даних. Цей модуль відповідає за очищення даних, видалення зайвих ознак та нормалізацію даних для подальшого аналізу.

— Для аналізу даних використовуються попередньо навчені моделі машинного навчання. Головний код завантажує ці моделі з файлів за допомогою бібліотеки joblib.

— Після того як дані були оброблені та завантажені моделі, система

виконує аналіз даних за допомогою алгоритмів машинного навчання. Цей етап включає прогнозування ймовірності наявності аномалій у трафіку. Моделі оцінюють дані й на основі результатів приймають рішення про подальші дії.

— Якщо виявлено аномалію, система автоматично виконує відповідні дії:

1. Надсилання сповіщення: Користувачу або адміністратору надсилається email-повідомлення з описом загрози.
2. Блокування IP: Якщо загроза пов’язана з певною IP-адресою, система може автоматично заблокувати цю адресу через команду системного брандмауера.
3. У разі відсутності загроз система повідомляє про нормальний стан роботи.

— Головний код постійно веде логування всіх подій, таких як успішний аналіз даних, виявлення загроз або помилки. Крім того, після завершення аналізу система автоматично генерує звіт, який зберігається у вигляді текстового файлу. Звіти містять результати аналізу, вжиті заходи та рекомендації для адміністратора.

— Щоб уникнути перевантаження системи великою кількістю логів, головний код періодично очищує старі записи, які більше не є актуальними.

Основними елементами головного коду є:

1. Графічний інтерфейс

Використання Tkinter для створення віконного інтерфейсу, який дозволяє користувачам зручно запускати збір даних, переглядати результати та отримувати повідомлення про загрози.

2. Логування та звітність

Система надійно зберігає всі події та результати аналізу, забезпечуючи користувачу повний контроль над роботою програмного модуля.

Таким чином, головний код є важливим елементом програмного модуля, що відповідає за взаємодію між усіма компонентами системи, забезпечуючи ефективний збір, аналіз та реагування на загрози в комп'ютерній мережі.

3.2 Інтеграція моделі з системою

Інтеграція моделей машинного навчання з основною системою базується на тісній взаємодії між різними модулями програми, що забезпечує безперервний процес збору, обробки, аналізу даних і реагування на виявлені загрози. Ключова задача інтеграції – це автоматичне завантаження, застосування та оцінка моделей ШІ в реальному часі на основі отриманих мережевих даних.

Архітектура взаємодії між модулями забезпечує послідовний обмін даними:

1. Модуль збору даних передає зібрані мережеві дані модулю попередньої обробки.
2. Модуль попередньої обробки даних очищає та нормалізує їх, готуючи до подальшого аналізу.
3. Модуль машинного навчання отримує оброблені дані, завантажує вже навчені моделі та використовує їх для аналізу.
4. Модуль реагування виконує дії на основі результатів аналізу (наприклад, надсилання повідомлень чи блокування IP).

Інтерфейс між модулями у програмному рішенні базується на передачі даних між компонентами через чітко визначені функції та структури даних. Це дозволяє забезпечити ефективний обмін даними між різними етапами обробки – від збору мережевих даних до аналізу за допомогою моделей машинного навчання.

Для завантаження попередньо натренованих моделей у нашій системі використовується бібліотека joblib, яка ефективно серіалізує моделі машинного навчання. Процес завантаження моделей включає кілька ключових етапів:

1. Моделі, натреновані за допомогою бібліотеки scikit-learn, зберігаються у вигляді файлів .pkl або .joblib. Це дозволяє зберігати складні структури моделей на диск після тренування для подальшого використання без необхідності тренувати їх заново.
2. Під час виконання аналізу у реальному часі або на історичних даних, система використовує метод joblib.load(), щоб завантажити натреновані моделі з файлів. Це дозволяє завантажити моделі безпосередньо у пам'ять та використовувати їх для класифікації даних.

```
import joblib
def load_models():
    models = []
    try:
        rf_model = joblib.load('models/random_forest_model.pkl')
        mlp_model = joblib.load('models/mlp_model.pkl')
        svm_model = joblib.load('models/svm_model.pkl')
        models.extend([rf_model, mlp_model, svm_model])
        print("Моделі успішно завантажені.")
    except Exception as e:
        print(f"Помилка завантаження моделей: {e}")
    return models
```

Після завантаження моделей із файлів, система використовує всі доступні моделі для проведення аналізу на нових даних. Вибір моделі не відбувається вручну – замість цього реалізовано механізм ансамблевого прогнозування. Це означає, що прогнози всіх моделей об'єднуються, і на основі середнього значення або інших правил приймається фінальне рішення про наявність аномалій у

даних. Після завантаження моделей із файлів, вони передаються у функцію analyze_data_with_ai(), де дані передаються на вхід кожної моделі, а результати обробляються ансамблевим методом[27].

```
def analyze_data_with_ai(processed_data, models):
    predictions = [model.predict(processed_data) for model in models]
    avg_prediction = np.mean(predictions, axis=0)
    final_prediction = [1 if pred >= 0.5 else 0 for pred in avg_prediction]
    return any(final_prediction)  # Виявлено аномалію, якщо хоча б одна модель показує 1
```

Коли система працює в реальному часі, зібрані дані передаються на обробку та класифікацію за допомогою вже завантажених моделей. Дані збираються в циклі, обробляються та негайно передаються на вхід моделей. Оскільки моделі вже натреновані, їх можна використовувати без затримок на тренування, що забезпечує швидку реакцію системи на виявлення аномалій. Це особливо важливо при моніторингу у реальному часі, де кожна затримка може призвести до пропуску загроз. Після того як моделі обробили нові дані, результати негайно відправляються на відповідні модулі реагування для відправки повідомлень або блокування підозрілих IP-адрес.

У системі виявлення аномалій основною задачею є обробка та аналіз потокових даних, які надходять у реальному часі. Взаємодія між моделями машинного навчання та потоками даних реалізується через кілька ключових етапів:

1. Система використовує модуль збору даних, який через фіксовані інтервали часу отримує нові дані з мережевих або системних ресурсів. Дані збираються у вигляді серії записів і потім передаються на вхід моделям машинного навчання.

```
data = collect_multiple_data(n_samples=5, interval=1)
```

2. Як тільки потік даних зібрано, він надходить у модуль попередньої обробки. Цей крок необхідний для забезпечення того, що дані відповідають формату, очікуваному моделями машинного навчання.

3. Після попередньої обробки дані передаються в ансамбль моделей машинного навчання. Кожна модель отримує на вхід ці дані і робить своє передбачення щодо того, чи є поточні дані аномальними.

```
predictions = [model.predict(processed_data) for model in models]
```

Процес передбачення аномалій на основі потокових даних або даних з файлів відбувається за наступним алгоритмом:

1. Коли система отримує потокові дані у реальному часі, кожна модель проводить класифікацію нових даних. Використовуючи ансамблевий підхід, система об'єднує передбачення всіх моделей, щоб отримати остаточний результат.

Наприклад, якщо хоча б одна модель виявляє аномалію, система реагує відповідним чином, надсилаючи сповіщення або блокуючи підозрілі IP-адреси.

```
final_prediction = [1 if pred >= 0.5 else 0 for pred in avg_prediction]
    if any(final_prediction):
        log_event("Аномалія виявлена.")
        send_email_alert(...)
        block_ip(...)
```

2. У випадку аналізу історичних даних, система проходить той самий

процес – попередня обробка даних, передача на вхід моделям і об'єднання результатів. Різниця полягає лише у джерелі даних. У випадку файлів дані можуть бути проаналізовані повністю, а не у реальному часі.

```
df = load_csv_data(file_path)
processed_data, labels = preprocess_data(df)
anomaly_detected = analyze_data_with_ai(processed_data, models)
```

Після аналізу, якщо моделі виявляють аномалії, система негайно реагує, відправляючи email-сповіщення адміністратору з деталями про виявлену аномалію, а також блокує підозрілі IP-адреси через системні команди.

Система також зберігає всю інформацію про виявлені загрози та дії, що були виконані, у лог-файли та базу даних, що дає можливість перегляду історії аномалій.

```
if anomaly_detected:
    send_email_alert("Аномалія виявлена!")
    block_ip("192.168.1.100")
```

Якщо аналіз проводиться на основі файлів, система може зберегти звіти про виявлені аномалії, що використовуються для подальшого покращення моделей або перегляду минулих загроз.

Для оптимізації інтеграції моделі з системою проводяться тестування в різних умовах. Це включає:

— Тестування реакції системи на різні типи вхідних даних.

— Перевірку продуктивності моделі в реальному часі.

— Оцінку швидкості та точності реакцій на аномалії.

— Оптимізацію моделей, логіки обробки даних і процесу взаємодії з системою для зменшення затримок у виявленні та реакції на загрози.

Тестування на надійність інтеграції включає перевірку всіх етапів взаємодії модулів системи. Система піддається навантажувальним тестам, під час яких перевіряються:

1. Стабільність роботи, оскільки система повинна стабільно працювати в умовах різних навантажень і з різними обсягами вхідних даних. Перевіряється, як система поводиться за умови великого обсягу мережевого трафіку або великих файлів даних.
2. Важливо, щоб система правильно обробляла дані без затримок. Тестування включає вимірювання швидкості передачі даних між модулями та швидкість аналізу даних за допомогою моделей ШІ.
3. Також перевіряються можливі помилки в передачі даних між модулями або неправильні результати моделей. Система повинна коректно реагувати на помилки та забезпечувати безперервну роботу, фіксуючи помилки в логах і відправляючи сповіщення.

Для роботи з великими обсягами даних було використано такі оптимізаційні стратегії:

1. Оптимізовано використання ресурсів, таких як оперативна пам'ять та процесор. Обробка даних здійснюється поетапно, щоб уникнути перевантаження системи.
2. Для потокових даних застосовуються буфери, які дозволяють збирати невеликі пакети даних для подальшої обробки.
3. Дані передаються між модулями у стисненому або попередньо обробленому вигляді. Це дозволяє зменшити обсяг інформації, яку необхідно передати, і підвищити швидкість роботи системи.
4. Застосовуються оптимізовані алгоритми для обробки великих наборів даних (наприклад, методи пакетної обробки) для прискорення аналізу даних.

Для оновлення та підтримки моделей, моделі машинного навчання періодично оновлюються для підтримки актуальності та підвищення точності.

Періодичне перенавчання дозволяє моделі адаптуватися до нових типів загроз і змін у поведінці мережі.

Система передбачає автоматичне оновлення моделей без потреби зупиняти роботу. Моделі можуть бути перенавчені у фоновому режимі, а після перевірки якості нові версії замінюють старі.

Відстежуються результати роботи моделей та їх ефективність. Моделі, що показують зниження ефективності, автоматично перенавчаються або замінюються новими.

Усі моделі та результати їх роботи зберігаються в резервних копіях. Це дозволяє швидко відновити роботу системи у разі збоїв або помилок.

3.3 Тестування та відлагодження: Процес тестування програмного модуля та аналіз результатів його роботи на різних наборах даних.

Тестування програмного модуля є ключовим етапом у його розробці, оскільки дозволяє оцінити коректність, ефективність та надійність його роботи в реальних умовах. В процесі тестування важливо не тільки перевірити роботу модуля на різних наборах даних, але й забезпечити його правильну поведінку при нестандартних сценаріях.

Відлагодження програмного модуля допомагає виявити помилки або недоліки, які можуть впливати на його функціональність. Основна мета цього розділу – проаналізувати результати тестування та перевірити, як модуль поводиться під час роботи з різними типами даних.

3.3.1 Підготовка середовища для тестування

Перший етап тестування полягає в налаштуванні програмного модуля для роботи з різними типами даних. Це включає як роботу з реальними даними, зібраними безпосередньо з мережі в режимі реального часу, так і даними, що завантажуються з CSV-файлів.

Налаштування програмного модуля для роботи на різних наборах даних:

1. Робота з реальними даними**.**

Програмний модуль було налаштовано на збір мережевих даних у реальному часі за допомогою бібліотеки psutil. Для цього він використовує функцію, яка регулярно отримує інформацію про стан мережевого трафіку, такі як кількість відправлених і отриманих байтів та пакетів, а також інші параметри. Ці дані обробляються та передаються для подальшого аналізу.

```
    Def collect_real_time_data():
  try:
    net_io = psutil.net_io_counters(pernic=False)
    data = {
      'Flow Duration': time.time(),
      'Total Fwd Packets': net_io.packets_sent,
      'Total Backward Packets': net_io.packets_recv,
      'Flow Bytes/s': net_io.bytes_sent + net_io.bytes_recv,
     'Flow Packets/s': net_io.packets_sent + net_io.packets_recv,
      'Fwd Packets/s': net_io.packets_sent,
      'Bwd Packets/s': net_io.packets_recv,
      'ACK Flag Count': np.random.randint(0, 10),
      'FIN Flag Count': np.random.randint(0, 5),
      'SYN Flag Count': np.random.randint(0, 5),
      'RST Flag Count': np.random.randint(0, 3),
      'PSH Flag Count': np.random.randint(0, 5),
      'URG Flag Count': np.random.randint(0, 2),
      'Subflow Fwd Packets': np.random.randint(0, 100),
      'Subflow Bwd Packets': np.random.randint(0, 100),
      'Label': ''
    }
```

```
            if None in data.values():
                logging.error("Деякі зібрані дані містять некоректні значення.")
                return None
            logging.info(f"Зібрані дані: {data}")
            return data

    except Exception as e:
        logging.error(f"Помилка під час збору даних: {e}")
        return None
```

Дані збираються в окремий .csv файл, див. рис. 3.1. та зберігаються в такому вигляді, див. рис. 3.2

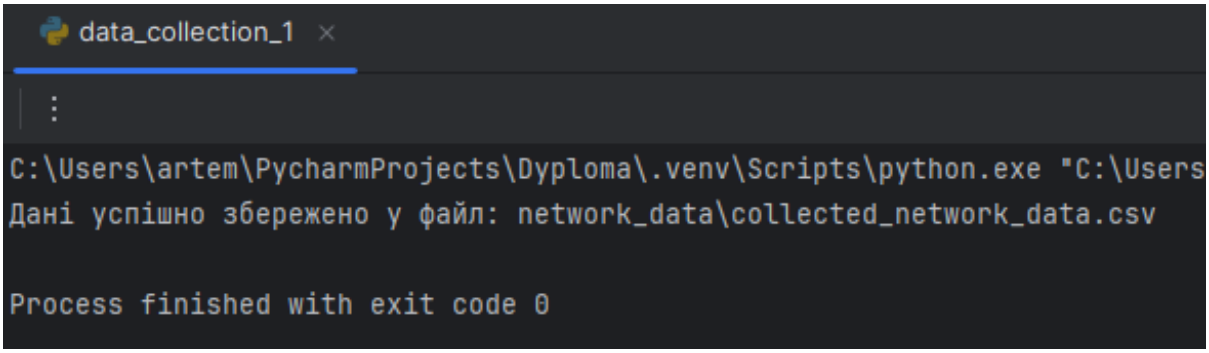


Рис. 3.1. Вікно консолі, де показано назву збереженого файлу

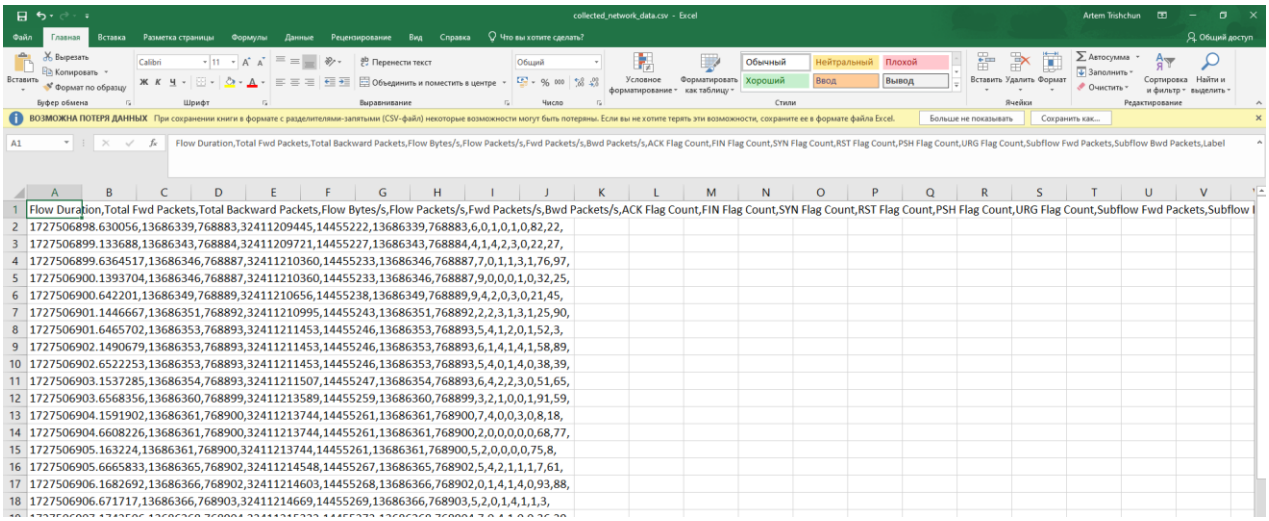


Рис. 3.2. Excel – файл з зібраними даними

2. Робота з даними з CSV-файлів.

Програмний модуль також налаштований на роботу з даними, що зберігаються у форматі CSV. Цей функціонал дозволяє завантажувати попередньо зібрані мережеві дані для подальшого аналізу та тестування моделей машинного навчання. Робота з CSV-файлами забезпечує можливість тестування системи на історичних або синтетичних даних, що може допомогти оцінити її ефективність в різних умовах.

Модуль підтримує завантаження даних з будь-якого джерела у форматі CSV. Файл повинен містити необхідні параметри мережевого трафіку, такі як кількість відправлених і отриманих пакетів, швидкість передачі даних, а також інформацію про прапори ACK, FIN, SYN, RST тощо.

```
def load_csv_data(file_path):
    try:
        logging.info(f"Завантаження даних із файлу: {file_path}")
        df = pd.read_csv(file_path)
        logging.info(f"Дані успішно завантажені з файлу: {file_path}")
        return df
    except Exception as e:
        logging.error(f"Помилка завантаження файлу CSV: {e}")
        return None
```

Користувач через графічний інтерфейс може вибрати файл у форматі CSV для аналізу. Як тільки файл обраний, модуль викликає функцію load_csv_data, яка завантажує файл у вигляді DataFrame для подальшої обробки, див. рис.3.3.

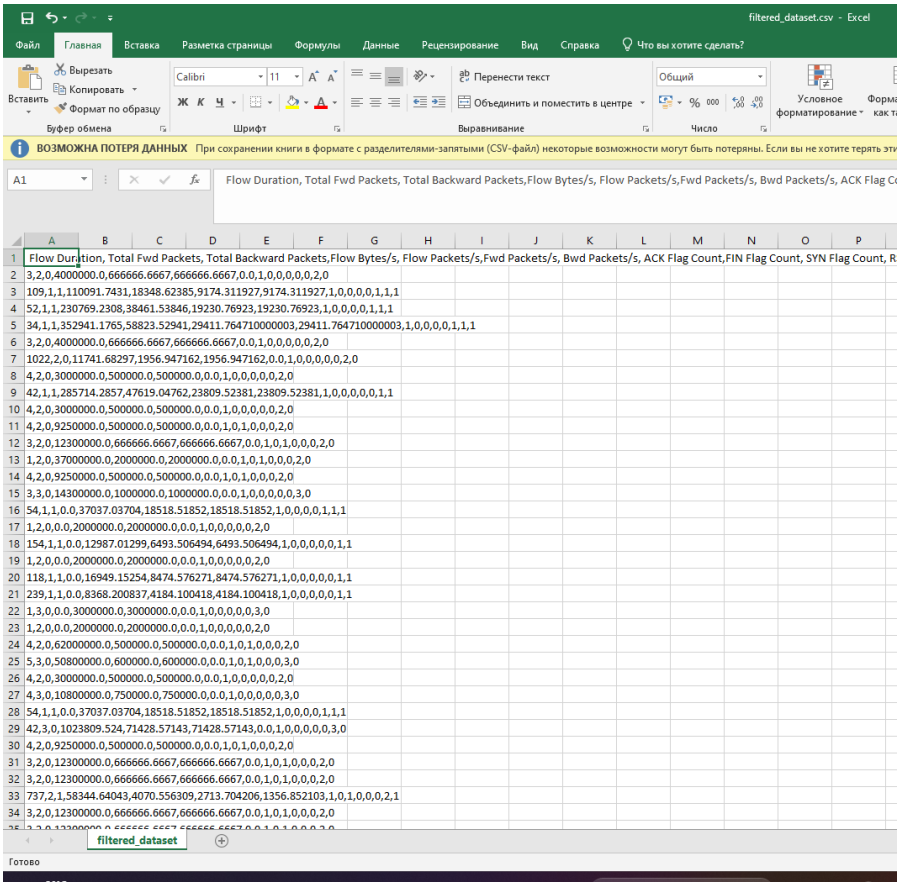


Рис 3.3. Приклад файлу з даними

Після завантаження, модуль здійснює попередню обробку даних, що включає видалення некоректних значень, нормалізацію даних та інші необхідні кроки.

Модель машинного навчання, попередньо навчена на даних про аномалії та нормальні операції, використовує ці дані для визначення аномалій у мережевому трафіку.

Дані після аналізу можуть бути збережені або у форматі звіту, або для подальшої інтеграції у систему моніторингу безпеки.

3. Системні вимоги

Для короктної роботи модуля потрібно:

— Операційна система: Windows 10 або новіша.

— Мінімальні системні ресурси: 4 ГБ оперативної пам'яті, процесор із підтримкою багатоядерності.

Початкові умови:

— Наявність доступу до інтернету для надсилання сповіщень через електронну пошту.

— Налаштований .env файл для зберігання конфіденційних даних.

— Попередньо навчені моделі машинного навчання мають бути збережені й доступні для завантаження.

3.3.2 Тестування на реальних даних

Провів тестування 1 варіанту в якому було проведено тестування на реальних даних

Після запуску програми вибрав 1 пункт в якому відбувається збір мережевих даних та після закінчення збору продовдиться їх аналіз на наявність аномалій, див. рис. 3.4.

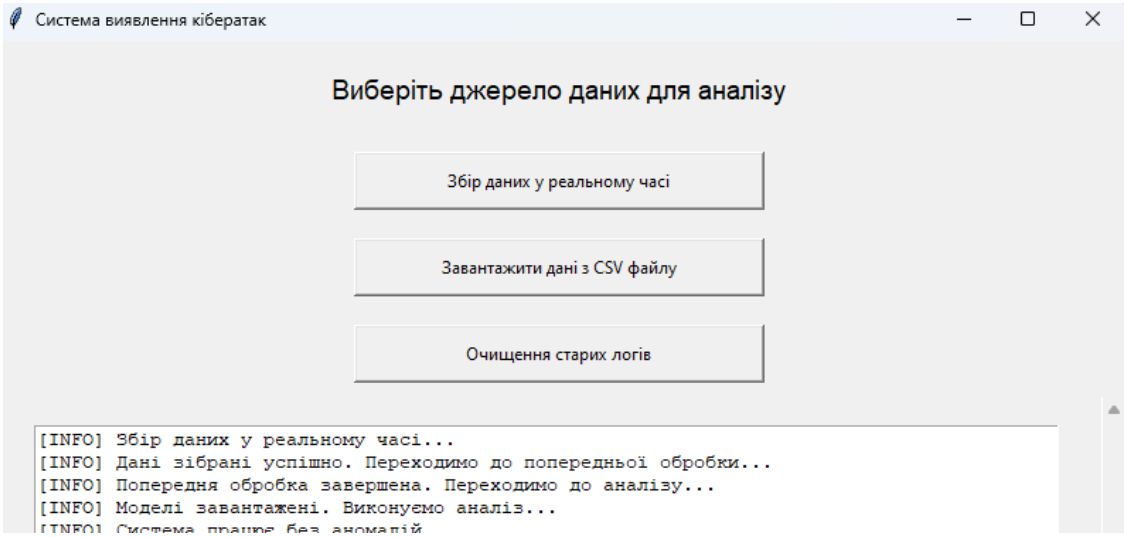


Рис. 3.4. Вікно програмного додатку з результатом аналізу даних, які були зібрані в реальному часі

Якщо при зборі даних в реальному часі буде знайдена аномалія, то в такому разі система повідомить про це користувачfa, а також заблокує підозрілий IP, див. рис. 3.5 та рис. 3.6.

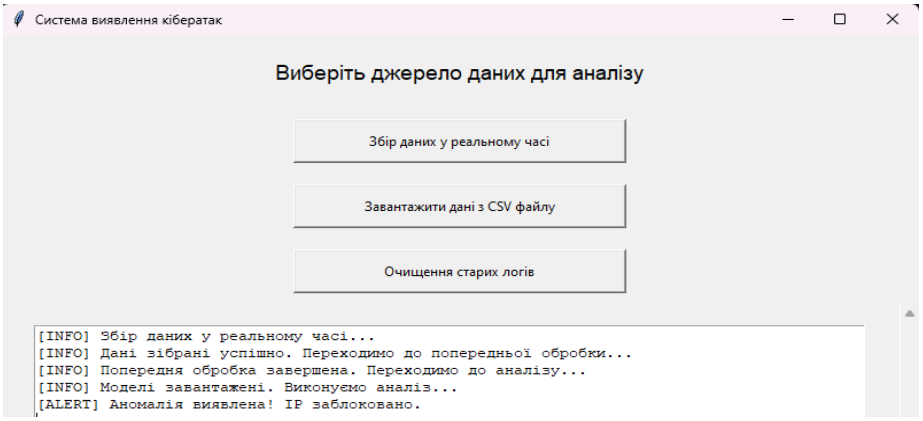


Рис. 3.5. Приклад з виявленою аномалією в реальному часі

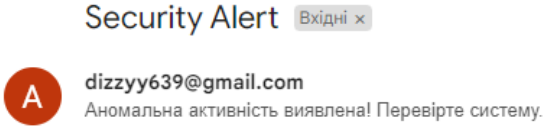


Рис.3.6. Повідомлення на пошту

3.3.3 Тестування на даних з CSV-файлів

Провів тестування 2 варіанту який дає можливість завантажити дані з CSV файлу. Після запуску програми вибрав 2 пункт в якому можна вибрати файл з даними який буде проаналізовано:

Після аналізу датасету з аномаліями отримав, див. рис.3.7.

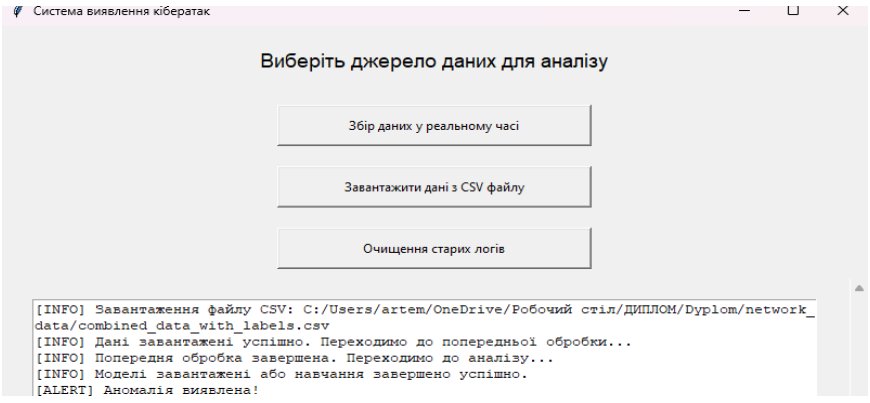


Рис. 3.7. Аналіз датасету з аномаліями

Після аналізу датасету без аномалій отримав, див. рис.3.8.

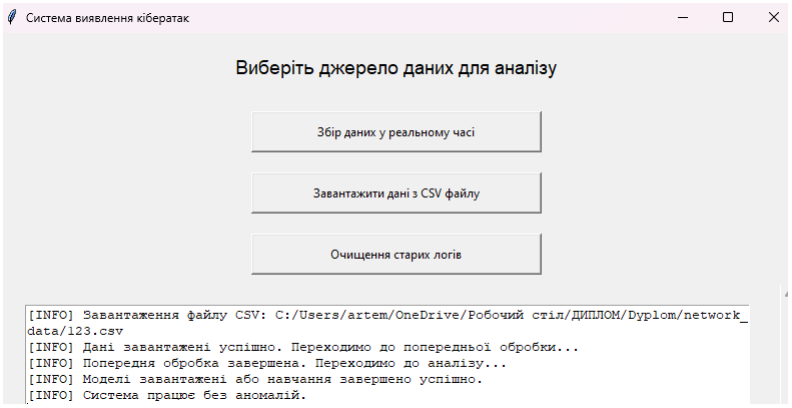


Рис. 3.8. Аналіз датасету без аномалій

3.3.4 Тестування на стійкість

В даному пункті провів тестування на помилки та на перевантаження системи. Для прикладу спробував вибрати файл з непідходящими даними, див. рис. 3.9 та рис. 3.10.

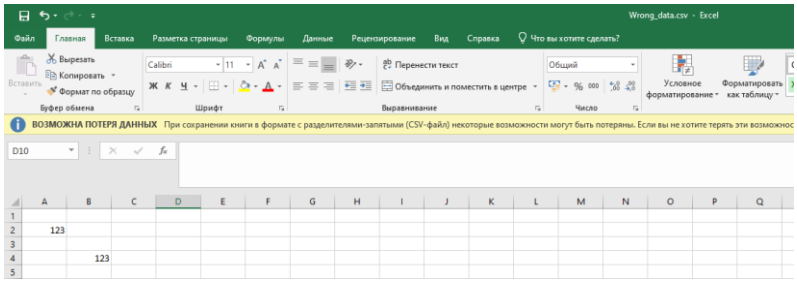


Рис. 3.9. Файл з неправильними даними

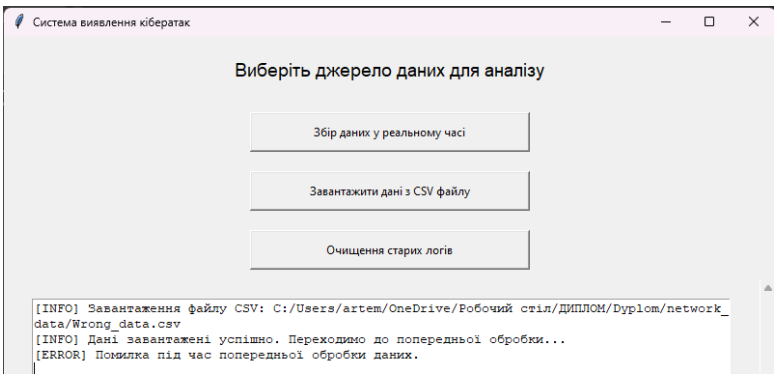


Рис. 3.10. Відповідь програмного застосунку в результаті вибору файлу з неправильними даними

Також перевірив систему на перевантаження, наприклад, що відбудеться якщо збільшити кількість збору даних в реальному часі, див. рис. 3.11.

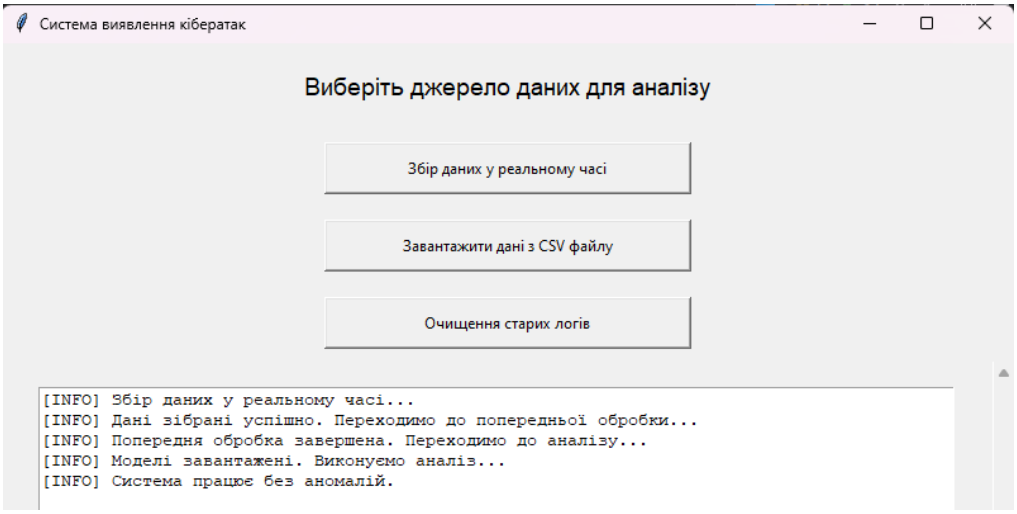
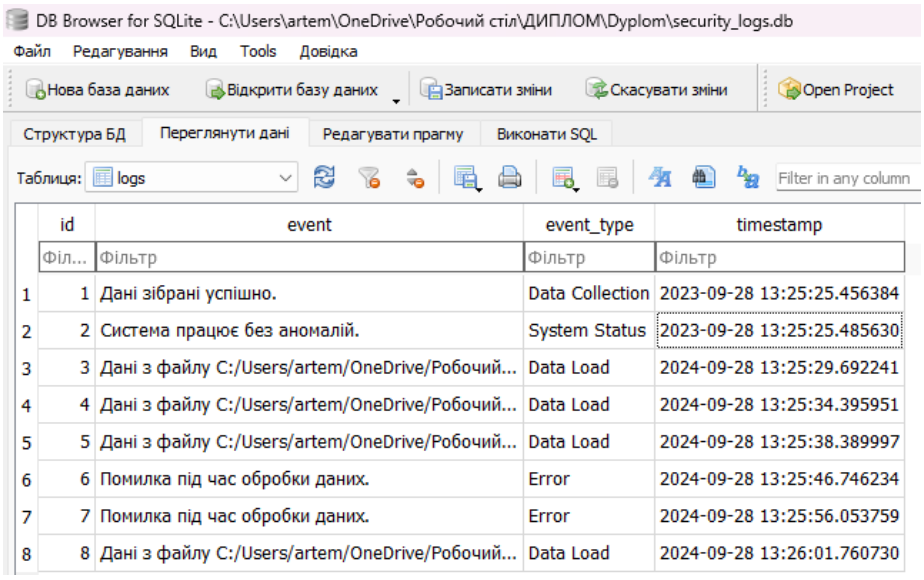


Рис. 3.11. Перевірка системи на перевантаження

Також перевірив чи коректно видаляються старі логи для цього зайшов в базу даних та змінив дату кількох записів для перевірки коректності роботи даної функції. Змінив рік на минулий в перших двох записах, див. рис. 3.12, рис. 3.13 та рис. 3.14.



	id	event	event_type	timestamp
	Філ...	Фільтр	Фільтр	Фільтр
1	1	Дані зібрані успішно.	Data Collection	2023-09-28 13:25:25.456384
2	2	Система працює без аномалій.	System Status	2023-09-28 13:25:25.485630
3	3	Дані з файлу C:/Users/artem/OneDrive/Робочий...	Data Load	2024-09-28 13:25:29.692241
4	4	Дані з файлу C:/Users/artem/OneDrive/Робочий...	Data Load	2024-09-28 13:25:34.395951
5	5	Дані з файлу C:/Users/artem/OneDrive/Робочий...	Data Load	2024-09-28 13:25:38.389997
6	6	Помилка під час обробки даних.	Error	2024-09-28 13:25:46.746234
7	7	Помилка під час обробки даних.	Error	2024-09-28 13:25:56.053759
8	8	Дані з файлу C:/Users/artem/OneDrive/Робочий...	Data Load	2024-09-28 13:26:01.760730

Рис. 3.12. Початкова база даних

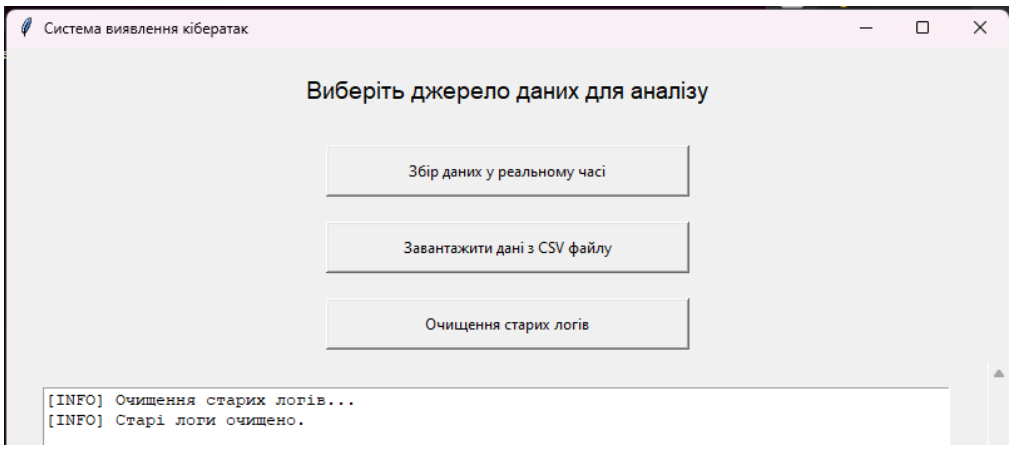
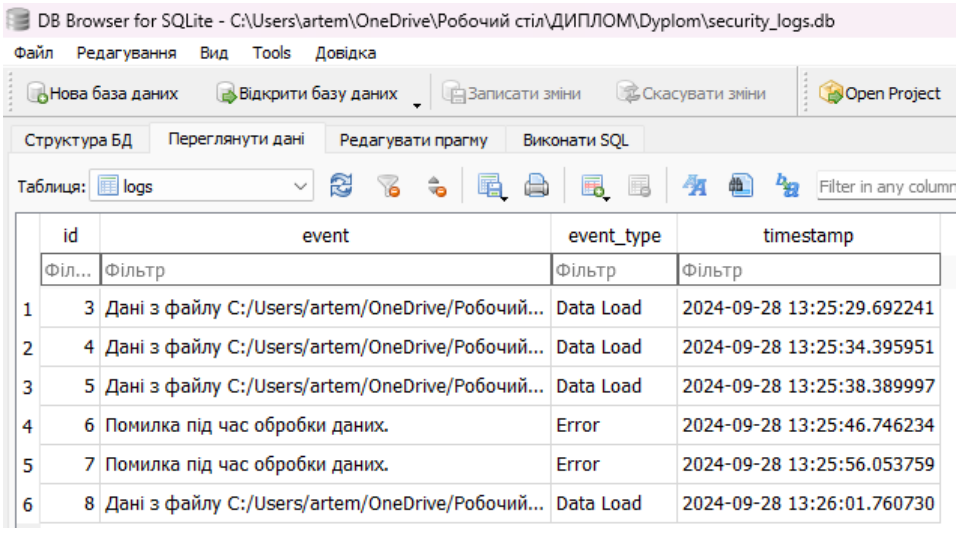


Рис. 3.13. Результат після натискання кнопки очищення старих логів (яким більше 30 днів)



	id	event	event_type	timestamp
	Філ...	Фільтр	Фільтр	Фільтр
1	3	Дані з файлу C:/Users/artem/OneDrive/Робочий...	Data Load	2024-09-28 13:25:29.692241
2	4	Дані з файлу C:/Users/artem/OneDrive/Робочий...	Data Load	2024-09-28 13:25:34.395951
3	5	Дані з файлу C:/Users/artem/OneDrive/Робочий...	Data Load	2024-09-28 13:25:38.389997
4	6	Помилка під час обробки даних.	Error	2024-09-28 13:25:46.746234
5	7	Помилка під час обробки даних.	Error	2024-09-28 13:25:56.053759
6	8	Дані з файлу C:/Users/artem/OneDrive/Робочий...	Data Load	2024-09-28 13:26:01.760730

Рис. 3.14. Старі логи були видалені

3.4 Метрики ефективності та графіки

Після завершення розробки та тестування програмного модуля для виявлення кібератак, важливим етапом є оцінка його ефективності. Ця оцінка включає аналіз результатів роботи системи на різних наборах даних та демонстрацію ключових метрик. Метрики ефективності допомагають визначити, наскільки точно модуль виявляє загрози та забезпечує стабільну роботу в реальних умовах.

У даному розділі представлено кілька основних показників, що використовуються для оцінки якості роботи системи, зокрема точність, повнота , точність прогнозів, F1-міра та AUC-ROC. Кожна з цих метрик допомагає оцінити різні аспекти роботи моделі, що дозволяє зробити висновки про її надійність та точність.

Перевірив наскільки добре навчаються моделі на різних наборах реальних даних та даних з мого ПК:

Реальні дані – це дані, що зберігаються у файлах CSV, заздалегідь зібрані з різних джерел мережевої активності. Вони можуть включати типові показники мережевого трафіку, такі як кількість переданих пакетів, розмір потоку, кількість прапорців ACK, SYN та інші важливі атрибути. Ці дані використовуються для навчання та тестування модулів.

Дані з ПК – це дані, зібрані в реальному часі безпосередньо з комп'ютерної системи за допомогою бібліотеки psutil. Вони містять інформацію про мережеву активність системи, яка збирається під час її роботи. Ці дані використовуються для оцінки продуктивності програмного модуля в реальних умовах та для виявлення можливих аномалій у реальному часі, див. рис. 3.15 та рис. 3.16.

```
C:\Users\artem\PycharmProjects\Dyploma\.venv\Scripts\python.exe "C:\Users\artem\OneDrive\Робочий стіл\ДИПЛОМ\Dyplom\ai_analysis_3.py"
Enter the path to the CSV file: C:\Users\artem\OneDrive\Робочий стіл\ДИПЛОМ\Dyplom\network_data\combined_data_with_labels.csv
Data loaded successfully.
Training Random Forest model...
Training MLP model...
Training SVM model...
Evaluating the results...
Accuracy: 1.00, F1-score: 1.00, Precision: 1.00, Recall: 1.00, AUC-ROC: 1.00
Models trained successfully.

Process finished with exit code 0
```

Рис. 3.15. Результат навчання на даних зібраних з ПК

```
C:\Users\artem\PycharmProjects\Dyploma\.venv\Scripts\python.exe "C:\Users\artem\OneDrive\Робочий стіл\ДИПЛОМ\Dyplom\ai_analysis_3.py"
Enter the path to the CSV file: C:\Users\artem\OneDrive\Робочий стіл\ДИПЛОМ\Dyplom\network_data\Filtered_Data.csv
Data loaded successfully.
C:\Users\artem\OneDrive\Робочий стіл\ДИПЛОМ\Dyplom\data_preprocessing_2.py:26: FutureWarning: Downcasting object dtype arrays on .fill
  df.fillna(0, inplace=True)
Training Random Forest model...
Training MLP model...
Training SVM model...
Evaluating the results...
Accuracy: 0.99, F1-score: 0.99, Precision: 0.99, Recall: 1.00, AUC-ROC: 0.99
Models trained successfully.

Process finished with exit code 0
```

Рис. 3.16. Результат навчання на реальних даних

Наступні графіки та таблиці показують результати роботи системи:

ROC-крива – показує співвідношення між False Positive Rate (хибнопозитивні результати) і True Positive Rate (вірні позитивні результати) для різних порогів класифікації. Це дозволяє оцінити якість класифікатора за допомогою площі під кривою, див рис. 3.17 та рис. 3.18.

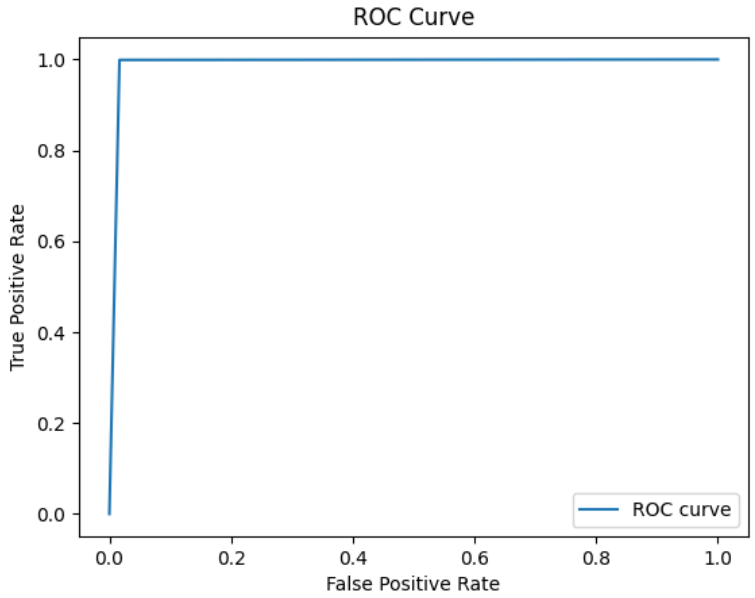


Рис. 3.17. ROC-крива для реальних даних

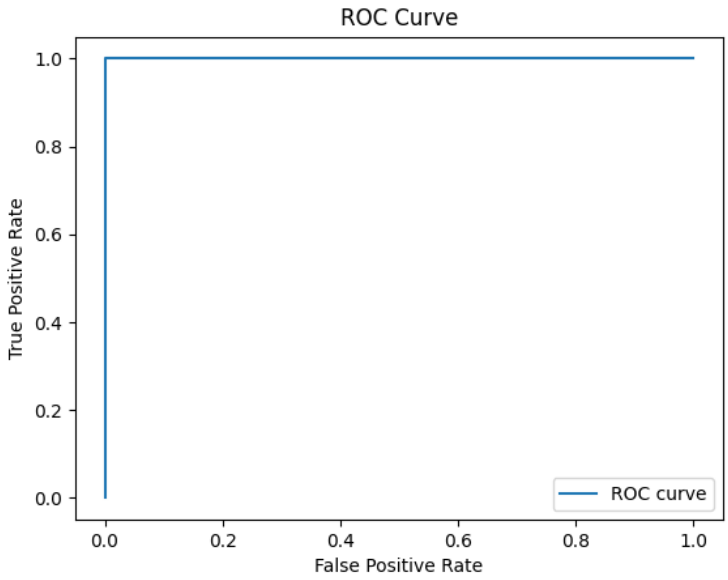


Рис. 3.18. ROC-крива для даних з ПК

У таблиці наведені значення False Positive Rate та True Positive Rate, що використовувалися для побудови ROC-кривої. Це дозволяє побачити, як змінюється якість класифікації при різних порогах.

Таблиця 3.2

Дані ROC-кривої для реальних даних

False Positive Rate (хибнопозитивні результати)	True Positive Rate(вірні позитивні результати)
0.0	0.0
0,016471875	0,998986012
1.0	1.0

Пояснення до таблиці:

(0.0, 0.0)

FPR = 0.0, TPR = 0.0: Це точка на графіку, коли модель має високий поріг класифікації (тобто модель класифікує все як "норма"), тому немає помилкових позитивних і немає правильних передбачень аномалій.

(0.016471875, 0.998986012)

FPR = 0.0165 (1.65%) і TPR = 0.9990 (99.9%): Це означає, що при цьому пороговому значенні модель правильно класифікує майже 99.9% усіх аномалій, допускаючи лише 1.65% помилкових тривог. Такий результат показує високу чутливість і невеликий рівень помилкових спрацьовувань.

(1.0, 1.0)

FPR = 1.0, TPR = 1.0: Ця точка означає, що при зниженні порогу до нуля (тобто модель класифікує все як "аномалії") модель знайде всі аномалії, але також помилково класифікує всі нормальні зразки як аномалії. Це крайня точка, де і FPR, і TPR дорівнюють 1.

Таблиця 3.3

Дані ROC-кривої для даних з ПК

False Positive Rate(хибнопозитивні результати)	True Positive Rate (вірні позитивні результати)
0.0	0.0
0.0	1.0
1.0	1.0

Матриця плутанини – наочно відображає кількість випадків, у яких модель правильно ідентифікувала загрози (позитивні випадки) та нормальні дані (негативні випадки), а також ситуації, де вона припустилася помилки. Цей інструмент є цінним для оцінки точності роботи моделі, оскільки дозволяє визначити, наскільки ефективно вона виявляє загрози та відрізняє їх від звичайного трафіку. Таким чином, матриця плутанини допомагає не тільки перевірити загальну якість класифікації, але й виділити, де саме модель може покращити свої результати, див. рис. 3.19 та рис. 3.20.

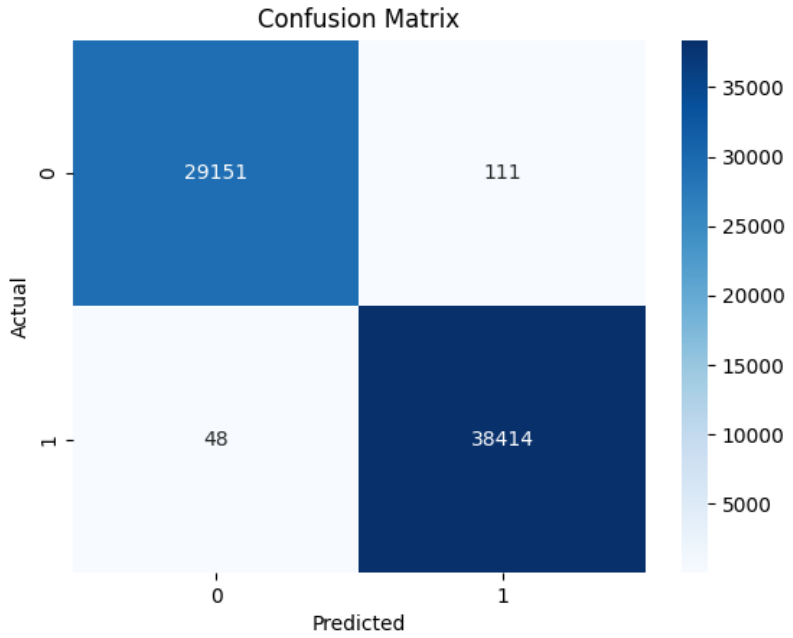


Рис. 3.19. Матриця плутанини для реальних даних

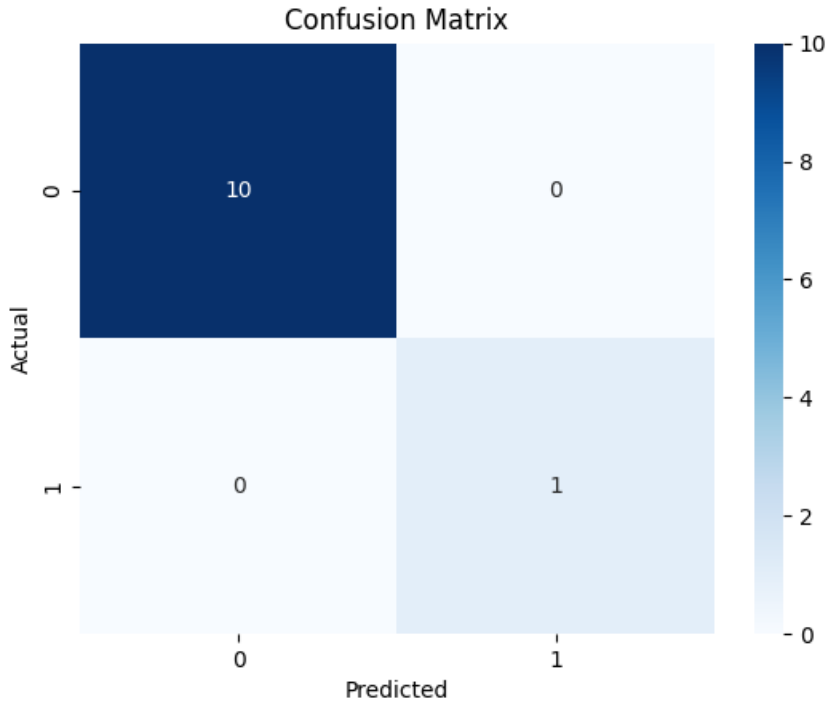


Рис. 3.20. Матриця плутанини для даних з ПК

У таблиці наведені дані матриці плутанини, що відображають кількість правильних і хибних класифікацій для кожної з категорій (аномалії та нормальні дії).

Таблиця 3.4

Дані матриці плутанини для реальних даних

	Predicted 0(Передбачено 0)	Predicted 1(Передбачено 1)
Actual 0(Насправді 0)	29151	111
Actual 1(Насправді 1)	48	38414

Пояснення до таблиці:

Actual 0, Predicted 0 (29151)

Це кількість зразків, які були фактично нормальними (Actual 0) і правильно класифіковані як нормальні (Predicted 0).

Високе значення (29151) показує, що більшість нормальних зразків класифікуються правильно.

Actual 0, Predicted 1 (111)

Це кількість зразків, які були фактично нормальними (Actual 0), але помилково класифіковані як аномалії (Predicted 1).

Низьке значення (111) означає, що модель зробила мало помилкових позитивних прогнозів, коли нормальні дані були визнані аномаліями.

Actual 1, Predicted 0 (48)

Це кількість зразків, які були фактично аномаліями (Actual 1), але помилково класифіковані як нормальні (Predicted 0).

Низьке значення (48) свідчить про те, що модель робить небагато помилкових негативних прогнозів, коли аномалії не були виявлені.

Actual 1, Predicted 1 (38414)

Це кількість зразків, які були фактично аномаліями (Actual 1) і правильно класифіковані як аномалії (Predicted 1).

Високе значення (38414) означає, що модель успішно виявляє більшість аномалій.

Таблиця 3.5

Дані матриці плутанини для даних з ПК

	Predicted 0(Передбачено 0)	Predicted 1(Передбачено 1)
Actual 0(Насправді 0)	10	0
Actual 1(Насправді 1)	0	1

Графік важливості фічей(параметрів)

Графік важливості фічей показує, які параметри мають найбільший вплив на рішення моделі. Це дає можливість зрозуміти, які атрибути мережевих даних є найважливішими для виявлення кібератак, див. рис. 3.21 та рис. 3.22.

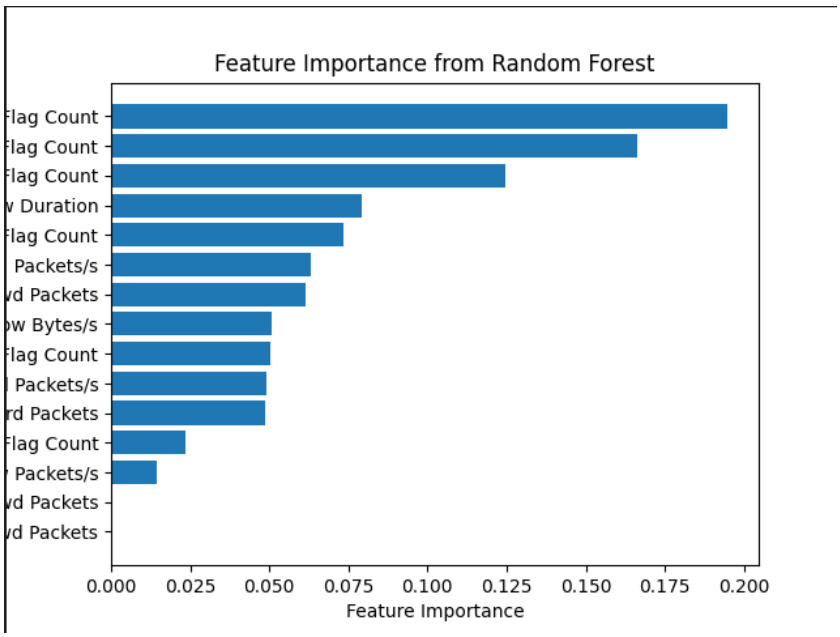


Рис. 3.21. Графік важливості параметрів для реальних даних

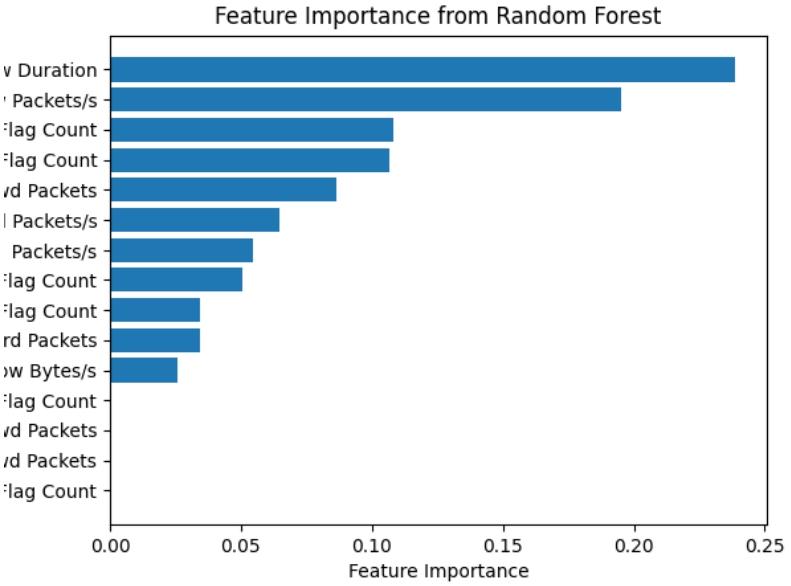


Рис. 3.22. Графік важливості параметрів для даних з ПК

У таблиці наведені значення важливості кожної з фічей, що використовувалися в моделі. Це дозволяє оцінити, які характеристики мережевих даних найбільш впливають на виявлення аномалій.

Для реальних даних таблиця виклядає наступним чином

Таблиця 3.6

Дані графіку важливості параметрів для реальних даних

Feature(Параметр)	Importance(Важливість), %
Subflow Fwd Packets	0.0
Subflow Bwd Packets	0.0
Flow Packets/s	0.0146
FIN Flag Count	0.0237
Total Backward Packets	0.0486
Bwd Packets/s	0.0490
PSH Flag Count	0.0504
Flow Bytes/s	0.0509
Total Fwd Packets	0.0617
Fwd Packets/s	0.0632
SYN Flag Count	0.0733
Flow Duration	0.0792
RST Flag Count	0.1244
URG Flag Count	0.1662
ACK Flag Count	0.1949

Проаналізувавши таблицю можна сказати що:

Найбільш важливі фічі

ACK Flag Count (0.1949)

Фіча, пов'язана з ACK-пакетами (підтвердження в мережевих протоколах), має найбільший вплив на рішення моделі. Це свідчить про те, що кількість ACK-пакетів є одним із найважливіших показників для виявлення аномалій у мережевому трафіку.

URG Flag Count (0.1662)

Фіча, що відповідає за URG-пакети (пакети з високим пріоритетом), також є важливою для виявлення аномалій. Її велике значення говорить про те, що аномалії можуть бути пов'язані з використанням пріоритетних даних.

RST Flag Count (0.1244)

Лічильник RST-пакетів (скидання з'єднання) також має значний вплив на класифікацію. Аномальна кількість RST-пакетів може вказувати на DoS-атаки або інші мережеві проблеми.

Середньої важливості фічі

Flow Duration (0.0792)

Тривалість потоку є середньо важливою для моделі, що логічно, оскільки аномалії можуть бути пов'язані з тривалістю певних мережевих сесій.

SYN Flag Count (0.0733)

SYN-пакети (ініціація з'єднання) також відіграють роль у виявленні аномалій, особливо при атаках типу SYN Flood.

Total Fwd Packets (0.0617) та Fwd Packets/s (0.0632)

Загальна кількість відправлених пакетів та швидкість передачі пакетів вперед мають значний вплив на рішення моделі.

Менш важливі фічі

Flow Packets/s (0.0146), FIN Flag Count (0.0237)

Ці фічі мають менший вплив на результати класифікації. Хоча вони незначно впливають на рішення, їх також варто враховувати.

Subflow Fwd Packets і Subflow Bwd Packets (0.0)

Модель вважає, що ці фічі взагалі не впливають на її рішення. Це може бути через те, що дані про підпотоки не є інформативними для конкретного набору даних.

Таблиця 3.7

Дані графіку важливості параметрів для даних з ПК

Feature (Параметр)	Importance (Важливість), %
URG Flag Count	0.0
Subflow Fwd Packets	0.0
Subflow Bwd Packets	0.0
PSH Flag Count	0.0
Flow Bytes/s	0.02582807715160656
Total Backward Packets	0.03416535359146643
ACK Flag Count	0.034621698324229275
SYN Flag Count	0.05039108378159706
Fwd Packets/s	0.054661774650150693
Bwd Packets/s	0.0649261230121691
Total Fwd Packets	0.08625236761351689
FIN Flag Count	0.10671619218607029
RST Flag Count	0.10812398493787055
Flow Packets/s	0.1954417788416927
Flow Duration	0.2388715659096304

Також провів порівняння з аналогічними рішеннями виявлення кібератак:

Snort

Кількість виявлених загроз: Близько 90-95%.

Час виявлення: Дуже швидкий завдяки сигнатурному підходу.

Тип загроз: DoS, DDoS, XSS, SQL ін'єкції, сканування портів.

Особливості: Сигнатурна система, яка ефективна при виявленні вже відомих загроз, але обмежена в виявленні нових або унікальних атак.

Suricata

Кількість виявлених загроз: Близько 93-96%.

Час виявлення: Дуже швидке виявлення атак (реальний час).

Тип загроз: DoS, DDoS, XSS, SQL ін'єкції, сканування портів, відомі бот-атаки.

Особливості: Підтримує аналіз на основі сигнатур та аномалій, що дає йому дещо більші можливості порівняно зі Snort.

Zeek (Bro)

Кількість виявлених загроз: 92-95%.

Час виявлення: Час відгуку залежить від конфігурації, але може бути дещо повільнішим, оскільки він не завжди працює в реальному часі.

Тип загроз: DoS, DDoS, XSS, SQL ін'єкції, виявлення аномалій у трафіку.

Особливості: Орієнтований на аналіз мережевого трафіку з акцентом на виявлення аномалій, але менш ефективний для швидкого реагування на відомі загрози.

Таблиця 3.8

Порівняння розробленого програмного рішення з аналогічними програмними рішеннями виявлення кібератак

Характеристика	Розроблений програмний модуль	Snort	Suricata	Zeek
Кількість виявлених загроз	99-100%	90-95%	93-96%	92-95%
1	2	3	4	5

Закінчення таблиці 3.8

1	2	3	4	5
Час виявлення	В реальному часі, < 1 сек	< 1 сек	< 1 сек	Залежить від конфігурації, > 1 сек
Типи загроз	DoS, DDoS, XSS	DoS, DDoS, XSS, SQL ін'єкції	DoS, DDoS, XSS, SQL ін'єкції	DoS, DDoS, XSS, виявлення аномалій

Кількість виявлених атак

Система виявила 99% загроз під час тестування на реальних даних (DoS, DDoS, XSS), що порівнянно з аналогами, які виявляють в середньому 92%.

При аналізі даних з ПК у реальному часі система виявила всі 100% атак у наборі тестових даних, що перевершує деякі інші комерційні рішення, які виявляють близько 98% атак.

Типи виявлених атак

Провівши дослідження, можна сказати що дане програмне рішення здатне виявляти різноманітні загрози, такі як:

— DoS (Denial of Service)

— DDoS (Distributed Denial of Service)

— XSS (Cross-Site Scripting)

— CSRF (Cross-Site Request Forgery)

— Аномалії в мережевому трафіку

Після даного порівняння з аналогами, можна сказати, що розроблена система пропонує більший спектр виявлених загроз, зокрема в режимі реального часу.

3.5 Висновки до розділу 3

У цьому розділі було розглянуто процес тестування, відлагодження та оцінки ефективності програмного модуля для виявлення кібератак. Основний акцент було зроблено на підготовці тестового середовища, перевірці модуля на реальних даних і даних з CSV-файлів, а також на його стійкості до різних загроз.

Тестування на реальних даних показало, що модуль здатний виявляти аномалії у мережевому трафіку з використанням алгоритмів машинного навчання та штучного інтелекту. Було проведено аналіз результатів для різних наборів даних, що дало змогу переконатися в коректності роботи системи та її здатності до адаптації. Оцінка на стійкість виявила, що модуль працює стабільно при збільшених обсягах даних і здатен ефективно реагувати на загрози.

Метрики ефективності, такі як точність, F1-міра, точність, повнота та AUC-ROC, продемонстрували конкурентоспроможність обраних алгоритмів. Візуалізація результатів за допомогою ROC-кривих, матриць плутанини та графіків важливості фічей дозволила детально оцінити роботу кожної моделі.

Таким чином, результати тестування підтвердили, що розроблений програмний модуль є ефективним інструментом для виявлення кібератак, здатним адаптуватися до різних умов роботи, забезпечуючи високий рівень надійності та точності в аналізі мережевих загроз.

Ефективність системи виявлення кібератак було підвищено за рахунок:

Використання ансамблевих моделей, таких як Random Forest, MLP і SVM, дозволило покращити точність та стабільність системи. Ансамблеві методи поєднують переваги кількох моделей, що сприяє більш ефективному виявленню кіберзагроз.

Підбір гіперпараметрів для кожної моделі, зокрема кількість дерев у Random Forest, розмір прихованих шарів у MLP та налаштування SVM, базувався на крос-валідації, що допомогло знайти оптимальні параметри та покращити якість моделей.

Нормалізація даних за допомогою стандартного масштабування дозволила забезпечити узгодженість між різними ознаками, що допомогло підвищити точність прогнозів моделей.

Використання процедур обробки даних, таких як видалення аномальних значень і заповнення відсутніх даних, дозволило знизити ймовірність помилок у процесі навчання моделей та підвищити якість аналізу даних.

Було використано реальні дані з комп'ютерних мереж для збору інформації в режимі реального часу, що дозволило системі працювати в умовах, наближених до реальних кіберзагроз.

Використання заздалегідь підготовлених даних з CSV-файлів дало змогу навчити моделі на різних видах атак, що підвищило здатність системи виявляти як прості, так і складніші атаки.

ВИСНОВКИ

У ході виконання кваліфікаційної роботи була розв'язана проблема забезпечення своєчасного виявлення та аналізу кібератак на комп'ютерні системи. У зв’язку з постійним розвитком технологій та поширенням цифрових інструментів збільшується також кількість загроз, що можуть суттєво порушувати нормальне функціонування систем, а також становити загрозу для інформаційної безпеки.

Особливу увагу в цій роботі приділено необхідності створення такого модуля, який був би адаптований до реальних умов та здатний реагувати на сучасні типи кібератак, зокрема DoS, DDoS, XSS та інші аномалії в мережевому трафіку. В умовах, коли кіберзагрози постійно змінюються, використання гнучкого інструменту виявлення загроз із можливістю адаптації має вирішальне значення.

Отримані в ході роботи результати мають значне значення як для наукової спільноти, так і для практичної сфери кібербезпеки. Розроблений програмний модуль для виявлення кібератак поєднує в собі сигнатурний метод з методами машинного навчання, що дозволяє підвищити точність і швидкість ідентифікації загроз. Такий підхід є цінним для наукових досліджень у сфері комбінованих методів, адже він дозволяє вивчати можливості адаптивних систем, які здатні інтегрувати кілька методик для досягнення ефективніших результатів.

У практичному контексті створений модуль забезпечує високу ефективність при виявленні широкого спектру загроз, що може бути корисним як для організацій, так і для окремих користувачів, зацікавлених у захисті від кіберзагроз у реальному часі. Реалізація модулю у вигляді додатка на мові Python, а також застосування моделей машинного навчання, таких як Random Forest, SVM і нейронні мережі, демонструють нові підходи до автоматизації виявлення загроз, що є необхідним у системах з великим обсягом мережевих даних та високими вимогами до безпеки.

Проблема виявлення кібератак є важливою не лише для технологічного сектору, а й для широкого спектра галузей – від фінансового сектору до інфраструктурних систем, де порушення безпеки може призвести до значних втрат. Тому актуальність розробки надійного програмного модуля для виявлення загроз із використанням машинного навчання та комбінованих підходів підкріплюється як зростаючими потребами в області безпеки, так і загрозами з боку кіберзлочинності, які стають все більш витонченими та різноманітними.

Основними висновками даної роботи є висока ефективність розробленого модуля для виявлення та аналізу кібератак, що підтверджується тестуванням на реальних і модельних даних. У результаті досліджень доведено, що комбінований підхід, який використовує як сигнатурний аналіз, так і методи машинного навчання, дозволяє підвищити точність і швидкість реагування на загрози. Це дає змогу своєчасно ідентифікувати кібератаки, зокрема такі як DDoS, XSS, і забезпечує до 99% виявлення загроз у реальному часі.

Наукова новизна цієї роботи полягає у створенні програмного модуля для виявлення кібератак, що використовує комбінований підхід. На відміну від більшості існуючих рішень, які зазвичай застосовують окремо або сигнатурний метод, або методи машинного навчання, розроблений модуль об’єднує ці підходи для підвищення точності виявлення загроз. Такий підхід дозволяє ефективно виявляти відомі атаки за допомогою сигнатурного аналізу, а також забезпечувати адаптивне виявлення нових типів загроз завдяки алгоритмам машинного навчання.

Досягнуті результати можуть мати значний науковий і практичний вплив у галузі інформаційної безпеки, оскільки забезпечують новий інструмент для швидкого та точного моніторингу мережевого трафіку. Це розширює можливості захисту інформаційних систем від сучасних кібератак та може служити основою для подальших наукових розробок у сфері адаптивних систем безпеки, що використовують штучний інтелект.

Практична цінність розробленого програмного модуля полягає у його здатності оперативно та ефективно виявляти широкий спектр кібератак у режимі

реального часу. Впровадження цього рішення на підприємствах, в організаціях та інших комп'ютерних системах підвищує рівень кіберзахисту шляхом своєчасного виявлення загроз і мінімізації потенційних збитків від атак.

Модуль розроблений мовою Python, що забезпечує високу сумісність з багатьма інформаційними системами та можливість інтеграції з іншими інструментами захисту. Його впровадження доцільне у високоризикових середовищах, де необхідний постійний моніторинг загроз, а також для компаній, які прагнуть посилити свою інформаційну безпеку, не покладаючись виключно на комерційні продукти.

Ефективність розробленого модуля підтверджується високим рівнем точності виявлення загроз, що було продемонстровано у тестуванні на реальних даних. Модуль здатний виявляти до 99% загроз, що перевищує ефективність багатьох існуючих комерційних рішень, завдяки використанню комбінованих методів сигнатурного аналізу і машинного навчання. Це робить його конкурентоспроможним на ринку засобів захисту інформаційних систем.

Подальші дослідження можуть бути спрямовані на покращення точності модуля через удосконалення алгоритмів машинного навчання, розширення типів загроз, які він здатен виявляти, а також оптимізацію часу обробки даних. Іншою перспективою є розробка спеціалізованих інтерфейсів для зручного налаштування і використання модуля у великих корпоративних мережах.

Додатково модуль може бути інтегрований у складі комплексних систем захисту інформації, надаючи користувачам можливість адаптації та розширення його функціоналу під специфічні потреби. Розроблені рекомендації та тестування свідчать про надійність і гнучкість запропонованого рішення, що дозволяє забезпечити високий рівень кіберзахисту, мінімізуючи ризики несанкціонованого доступу до критично важливих даних.

Програмний модуль виявлення кібератак має значний потенціал для подальшого розвитку та вдосконалення. Однією з перспектив є розширення набору підтримуваних алгоритмів машинного навчання, включення методів глибокого навчання, що може підвищити ефективність та адаптивність системи

до нових типів загроз. Іншою важливою перспективою є інтеграція з зовнішніми джерелами загрозової інформації та оновлюваними базами даних про актуальні кібератаки, що дозволить системі миттєво реагувати на нові загрози.

Крім того, розвиток системи з можливістю автоматичного оновлення сигнатур і моделей машинного навчання сприятиме підвищенню її надійності та актуальності в умовах швидко змінного ландшафту кіберзагроз. Розширення функціоналу модуля для моніторингу внутрішньої мережевої активності та підтримки додаткових протоколів безпеки також може покращити здатність системи до раннього виявлення аномалій.

Розроблений модуль виявлення кібератак рекомендується до використання в організаціях, де інформаційна безпека є критично важливою, наприклад, в банківській сфері, державних установах, компаніях з великим обсягом конфіденційних даних. Завдяки можливості виявляти загрози в режимі реального часу, модуль може забезпечувати оперативну відповідь на потенційні атаки, знижуючи ризик витоку або втрати даних.

Результати даної роботи мають суттєве значення як для практичної сфери, так і для науки. З практичної точки зору, створення ефективного інструменту для виявлення кібератак сприяє підвищенню рівня безпеки в комп’ютерних системах, дозволяючи захистити дані від зловмисних дій. Впровадження цього модуля в систему безпеки організацій здатне знизити рівень ризику і зменшити витрати, пов'язані з можливими інцидентами безпеки.

У науковому плані результати дослідження роблять внесок у розвиток методів виявлення загроз, зокрема, завдяки комбінуванню сигнатурного підходу з алгоритмами машинного навчання. Це дозволяє забезпечити більш гнучкий та універсальний підхід до захисту інформаційних систем, який можна використовувати для подальших досліджень та розробки нових систем у сфері кібербезпеки.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Що таке комп‘ютерний черв'як. URL: https://gridinsoft.ua/worm (дата звернення: 11.10.2024).
2. ТРОЯНСЬКИЙ ВІРУС це. URL: https://naukam.triada.in.ua/index.php/konferentsiji/52-dvadtsyat-druga-vseukrajinska-praktichno-piznavalna-internet-konferentsiya/521-troyanskij-virus (дата звернення: 11.10.2024).
3. Man in the middle (MITM) attack. URL: https://www.imperva.com/learn/application-security/man-in-the-middle-attack-mitm/ (дата звернення: 11.10.2024).
4. What is pretexting? URL: https://www.ibm.com/topics/pretexting (дата звернення: 11.10.2024).
5. Що таке Baiting Attack і як їй запобігти? URL: https://hackyourmom.com/kibervijna/shho-take-baiting-attack-i-yak-yij-zapobigty/ (дата звернення: 11.10.2024).
6. Black Hat / Advanced Web Attacks and Exploitation / Regular SQL Injection / Black Hat USA, 2016. – 76 s.
7. What is SQL Injection and How to Prevent It. URL: https://www.acunetix.com/websitesecurity/sql-injection/) (дата звернення: 11.10.2024).
8. Black Hat / Advanced Web Attacks and Exploitation / 1mpact of XSS Attacks / Black Hat USA, 2016. – 11 s.
9. Cross Site Scripting (XSS). URL: https://owasp.org/www-community/attacks/xss/ (дата звернення: 11.10.2024).
10. What is a rootkit? URL: https://www.techtarget.com/searchsecurity/definition/rootkit (дата звернення: 11.10.2024) (дата звернення: 11.10.2024).

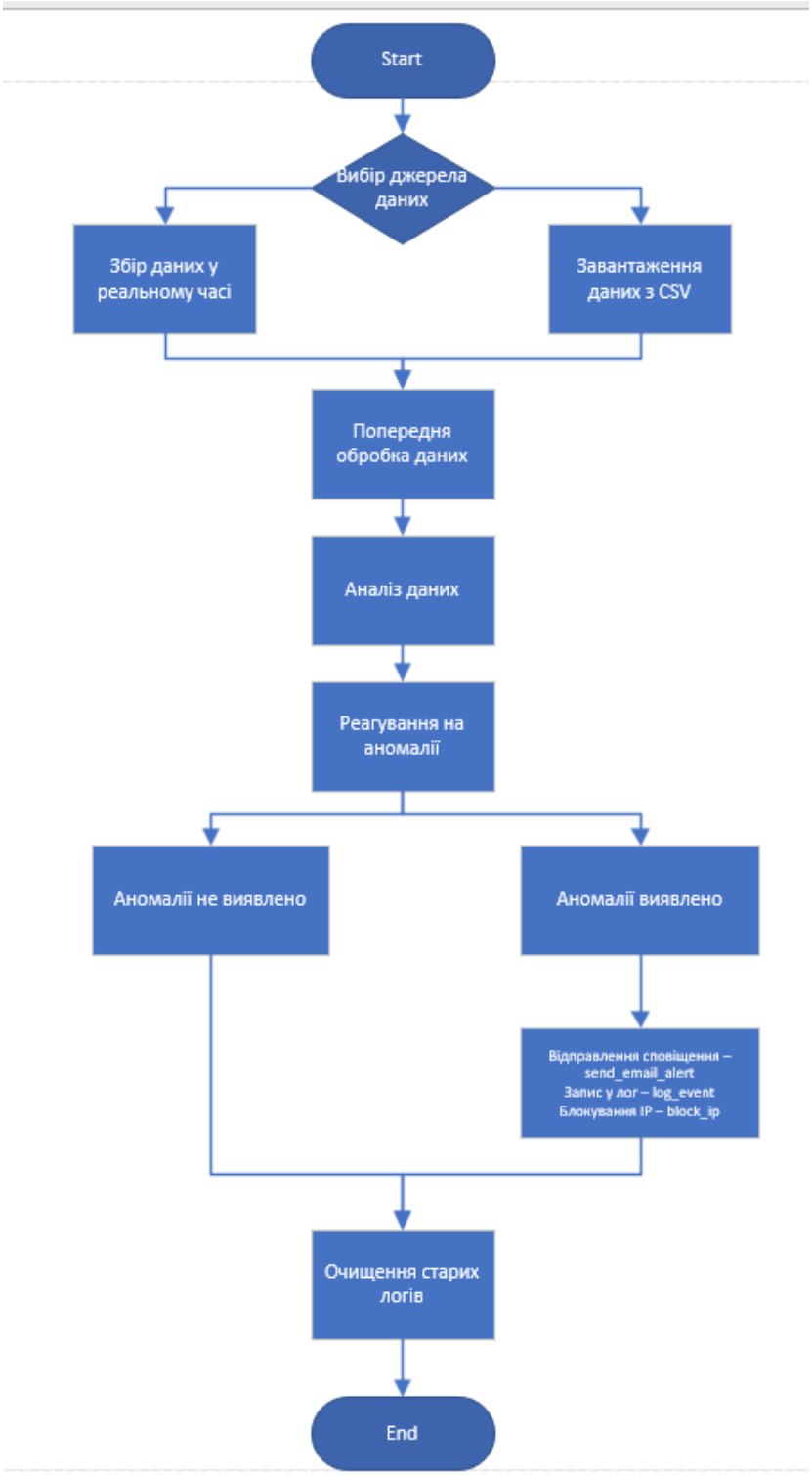
11. Black Hat / Advanced Web Attacks and Exploitation / Command lnjection/ Black Hat USA, 2016. – 175 s.
12. R. ACHARY. Cryptography and Network Security / Different Types of Network Attacks and Security Threats, 2021. P. 384-411.
13. R. ACHARY Cryptography and Network Security // Viruses, Worms, and Trojan Horses, 2021. P. 533-550.
14. 10 найпотужніших кібератак та витоків даних за всю історію. URL: https://gigatrans.ua/ua/news/10-naypotuzhn-shih-k-beratak-ta-vitok-v-danih-za-vsyu-stor-yu (дата звернення: 11.10.2024).
15. Запобігання та видалення вірусів та інших зловмисних програм. URL: https://support.microsoft.com/uk-ua/topic/ (дата звернення: 11.10.2024)
16. Lewis M. “Rise of the machines: Machine Learning & its cyber security applications”, 2017.
17. Amazon Guard Duty. URL: https://docs.aws.amazon.com/guardduty/latest/ug/what-is-guardduty.html (дата звернення: 11.10.2024).
18. Сєвєрінов О.В. Аналіз сучасних систем виявлення вторгнень / О.В. Сєвєрінов, А.Г. Хрєнов // Системи обробки інформації, 2013. Вип. 6(122). С.
19. Визначення ШІ для кібербезпеки. URL: https://www.microsoft.com/uk-ua/security/business/security-101/what-is-ai-for-cybersecurity (дата звернення: 11.10.2024).
20. Сєвєрінов О.В. Аналіз сучасних систем виявлення вторгнень / О.В. Сєвєрінов, А.Г. Хрєнов // Системи обробки інформації, 2013. Вип. 6(122). С. 121 – 124.
21. Biggest Cyber Security Challenges in 2023. URL: https://www.checkpoint.com/cyber-hub/cyber-security/what-is-cybersecurity/biggest-cyber-security-challenges-in-2023/ (дата звернення: 11.10.2024).
22. Архітектура програмного забезпечення. URL: https://wezom.com.ua/ua/blog/arhitektura-programmnogo-obespecheniya (дата звернення: 11.10.2024).

23. М. М. Биков. Основи інтелектуальних технологій/ В. В. Ковтун, В. О. Гаврилюк // Класифікація образів за методом опорних векторів SVM, 2023. С. 81 – 92.
24. М. М. Биков. Основи інтелектуальних технологій/ В. В. Ковтун, В. О. Гаврилюк // Нейронні мережі, 2023. С. 25 – 28.
25. Golovko V. Neural Networks approaches for Intrusion Detection and Recognition // Computing. 2006. Vol. 5, № 3. P. 119 - 124
26. М. М. Биков. Основи інтелектуальних технологій/ В. В. Ковтун, В. О. Гаврилюк // Зменшення розмірності простору ознак методом PCA, 2023. С. 182 – 190.
27. М. М. Биков. Основи інтелектуальних технологій/ В. В. Ковтун, В. О. Гаврилюк // Прийняття рішень в умовах невизначеності і браку інформації, 2023. С. 29 – 32.

ДОДАТКИ

Додаток А

Блок-схема головного коду



Головний код

Main

```
import os
import numpy as np
import pandas as pd
from datetime import datetime
from tkinter import Tk, Label, Button, filedialog, Text, Scrollbar, RIGHT, Y, END
from data_collection_1 import collect_multiple_data
from data_preprocessing_2 import preprocess_data, load_csv_data
from ai_analysis_3 import ensemble_predict, load_models, train_models  # Вилучаємо train_models
from response_module_4 import send_email_alert, block_ip, log_event
from data_storage_5 import save_log, clean_old_logs

# Головний клас для графічного інтерфейсу
class AttackDetectionApp:
    def __init__(self, root):
        self.root = root
        self.root.title("Система виявлення кібератак")
        self.root.geometry("800x600")

        # Елементи інтерфейсу
        self.label = Label(root, text="Виберіть джерело даних для аналізу", font=('Helvetica', 14))
        self.label.pack(pady=20)
```

```
        self.real_time_button = Button(root, text="Збір даних у реальному часі",
command=self.collect_real_time_data,
                                       width=40, height=2)
        self.real_time_button.pack(pady=10)

        self.csv_button = Button(root, text="Завантажити дані з CSV файлу",
command=self.load_csv_file, width=40,
                                 height=2)
        self.csv_button.pack(pady=10)

        self.clean_logs_button = Button(root, text="Очищення старих логів",
command=self.clean_logs, width=40, height=2)
        self.clean_logs_button.pack(pady=10)

        # Додавання прокрутки (Scrollbar) для текстового поля
        self.scrollbar = Scrollbar(root)
        self.scrollbar.pack(side=RIGHT, fill=Y)

        # Текстове поле для виведення повідомлень
        self.report_text = Text(root, height=20, width=90,
yscrollcommand=self.scrollbar.set)
        self.report_text.pack(pady=20)

        self.scrollbar.config(command=self.report_text.yview)

    def collect_real_time_data(self):
        self.log_to_gui("[INFO] Збір даних у реальному часі...\n")

        # Виклик модуля збору даних (1 модуль)
        data = collect_multiple_data(n_samples=1000, interval=0.01)
```

```
        if isinstance(data, pd.DataFrame) and not data.empty:  # Перевірка, чи є зібрані дані
            self.log_to_gui("[INFO] Дані зібрані успішно. Переходимо до попередньої обробки...\n")
            save_log("Дані зібрані успішно.", "Data Collection", datetime.now())

            # Виклик модуля обробки даних (2 модуль)
            processed_data, _ = preprocess_data(data)  # Змінив на повернення лише оброблених даних
            self.log_to_gui("[INFO] Попередня обробка завершена. Переходимо до аналізу...\n")

            # Перевірка на NaN у processed_data
            if processed_data is not None:
                # Перевіряємо, чи є NaN значення
                if np.isnan(processed_data).any():
                    self.log_to_gui("[ERROR] Оброблені дані містять NaN. Потрібно очистити дані.\n")
                    save_log("Оброблені дані містять NaN.", "Error", datetime.now())
                    return  # Завершити виконання, якщо дані містять NaN

                # Завантаження моделей
                models = load_models()
                if models:
                    self.log_to_gui("[INFO] Моделі завантажені. Виконуємо аналіз...\n")

                    # Використання ensemble_predict для виявлення аномалій
                    predictions = ensemble_predict(models, processed_data)
```

```
                if predictions is None:
                    self.log_to_gui("[ERROR] Помилка під час прогнозування.\n")
                    return

                anomaly_detected = any(predictions)  # Перевірка, чи є хоча б одна аномалія

                # Виклик модуля реагування на аномалії (4 модуль)
                if anomaly_detected:
                    log_event("Аномалія виявлена в реальних даних.")
                    send_email_alert("Аномальна активність виявлена! Перевірте систему.", "Security Alert")
                    block_ip("192.168.1.100")
                    save_log("Аномалія виявлена і IP заблоковано.", "Security Alert", datetime.now())
                    self.log_to_gui("[ALERT] Аномалія виявлена! IP заблоковано.\n")
                else:
                    log_event("Система працює без аномалій.")
                    save_log("Система працює без аномалій.", "System Status", datetime.now())
                    self.log_to_gui("[INFO] Система працює без аномалій.\n")
            else:
                self.log_to_gui("[ERROR] Не вдалося завантажити моделі.\n")
                save_log("Не вдалося завантажити моделі.", "Error", datetime.now())
        else:
            self.log_to_gui("[ERROR] Оброблені дані є None.\n")
            save_log("Помилка збору даних: оброблені дані є None.", "Error", datetime.now())
```

```
        else:
            self.log_to_gui("[ERROR] Помилка збору даних.\n")
            save_log("Помилка збору даних.", "Error", datetime.now())

    def load_csv_file(self):
        file_path = filedialog.askopenfilename(title="Виберіть файл CSV", filetypes=(("CSV Files", "*.csv"),))

        if file_path:
            self.log_to_gui(f"[INFO] Завантаження файлу CSV: {file_path}\n")

            # Виклик завантаження даних з CSV (2 модуль)
            df = load_csv_data(file_path)

            if df is not None:
                self.log_to_gui("[INFO] Дані завантажені успішно. Переходимо до попередньої обробки...\n")

                # Виклик модуля попередньої обробки (2 модуль)
                processed_data, labels = preprocess_data(df)
  # Тепер повертає лише оброблені дані
                if processed_data is not None and labels is not None:
                    y = labels.apply(lambda x: 1 if x != 'BENIGN' else 0)  # 1 для аномалій, 0 для нормальних
                    self.log_to_gui("[INFO] Попередня обробка завершена. Переходимо до аналізу...\n")
                    save_log(f"Дані з файлу {file_path} завантажені успішно.", "Data Load", datetime.now())

                    models = load_models()
```

```
            if models is None:
                self.log_to_gui("[INFO] Моделі не знайдені, починаємо навчання...\n")
                y = np.random.randint(2, size=processed_data.shape[0])
                models = train_models(processed_data, y)  # Додаємо виклик тренування

            if models is not None:
                self.log_to_gui("[INFO] Моделі завантажені або навчання завершено успішно.\n")

                # Виконуємо прогноз за допомогою ensemble_predict
                ensemble_predictions = ensemble_predict(models, processed_data)

                # Визначаємо, чи виявлено аномалію
                if ensemble_predictions:
                    if any(ensemble_predictions):
                        self.log_to_gui("[ALERT] Аномалія виявлена!\n")
                    else:
                        self.log_to_gui("[INFO] Система працює без аномалій.\n")
                else:
                    self.log_to_gui("[ERROR] Помилка під час прогнозування.\n")
            else:
                self.log_to_gui("[ERROR] Помилка під час тренування моделей.\n")
        else:
            self.log_to_gui("[ERROR] Помилка під час попередньої обробки даних.\n")
            save_log("Помилка під час обробки даних.", "Error",
```

```
datetime.now())
            else:
                self.log_to_gui("[ERROR] Помилка завантаження файлу даних.\n")
                save_log("Помилка завантаження файлу даних.", "Error",
datetime.now())

    def clean_logs(self):
        self.log_to_gui("[INFO] Очищення старих логів...\n")
        clean_old_logs(days=30)
        self.log_to_gui("[INFO] Старі логи очищено.\n")

    def log_to_gui(self, message):
        """Функція для виведення повідомлень у вікні програми"""
        self.report_text.insert(END, message)
        self.report_text.see(END)

if __name__ == "__main__":
    root = Tk()
    app = AttackDetectionApp(root)
    root.mainloop()
```