Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра захисту інформації

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА НА ТЕМУ:

«Метод та засіб аналізу мережевих подій для SIEM-системи»

Виконав: студент 2 курсу групи 1БС-23м
спеціальності 125 Кібербезпека та захист інформації
_____ Данило ГОНЧАР

Керівник: к. т. н., доцент каф. ЗІ
_____ Віталій ЛУКІЧОВ
«13» 12 2024 р.

Рецензент: к.т.н., доц. каф. ПЗ
_____ Володимир МАЙДАНЮК
«13» 12 2024 р.

Допущено до захисту
Завідувач кафедри ЗІ
д. т. н., проф.
_____ Володимир ЛУЖЕЦЬКИЙ
«13» 12 2024 р.

Вінниця 2024

Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра захисту інформації
Рівень вищої освіти II (магістерський)
Галузь знань – 12 Інформаційні технології
Спеціальність – 125 Кібербезпека та захист інформації
Освітньо-професійна програма – Безпека інформаційних і комунікаційних систем

ЗАТВЕРДЖУЮ
Завідувач кафедри
ЗІ,д. т. н., проф.
_____ Володимир
ЛУЖЕЦЬКИЙ
«18» 09 2024 року

ЗАВДАННЯ
НА МАГІСТЕРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ
СТУДЕНТУ

Гончару Данилу Андрійовичу

1. Тема роботи: «Метод та засіб аналізу мережевих подій для SIEM-системи»
   керівник роботи: Лукічов Віталій Володимирович, к. т. н., доцент кафедри ЗІ,затверджений наказом ректора ВНТУ від 17 вересня 2024 року №310
   Строк подання студентом роботи 14 грудня 2024 року
2. Вихідні дані до роботи:
   - збір мережевих подій;
   - аналіз мережевих подій;
   - реагування на події;
   - інтеграція з SIEM-системами.
3. Зміст текстової частини: Вступ. 1. Аналіз інформаційних джерел. 2. Метод аналізу мережевих подій для SIEM системи. 3. Засіб аналізу мережевих подій для SIEM системи 4. Економічна частина. Висновки. Список використаних джерел. Додатки.
6. Перелік ілюстративного матеріалу: Схема роботи SIEM систем подій (плакат А4). Підхід машинного навчання (плакат А4).Загальний алгоритм роботи засобу (плакат А4). Схема роботи з SIEM системою (плакат А4). Графічний інтерфейс користувача засобу (плакат А4). Результати тестування засобу (плакат А4).

7. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завд при
1	Віталій ЛУКІЧОВ, к.т.н., доц. каф. ЗІ	10.09.24	
2	Віталій ЛУКІЧОВ, к.т.н., доц. каф. ЗІ	16.09.24	
3	Віталій ЛУКІЧОВ, к.т.н., доц. каф. ЗІ	14.10.24	
4	Ольга РАТУШНЯК, к.т.н., доц. каф. ЕПВМ	11.11.24	

8. Дата видачі завдання

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів магістерської кваліфікаційної роботи	Строк виконання етапів роботи	Прим
1	Аналіз завдання. Вступ	01.09.2024 – 10.09.2024	
2	Аналіз літературних джерел за напрямком магістерської кваліфікаційної роботи	10.09.2024 – 15.09.2024	
3	Науково-технічне обгрунтування	16.09.2024 – 22.09.2024	
4	Аналіз предметної області та формування вимог до програмного засобу	23.09.2024 – 29.09.2024	
5	Розробка архітектури програмного засобу	30.09.2024 – 12.10.2024	
6	Програмна реалізація засобу	14.10.2024 – 02.11.2024	
7	Тестування розробленого засобу	03.11.2024 – 10.11.2024	
8	Розробка розділу економічного обгрунтування доцільності розробки	11.11.2024 – 17.11.2024	
9	Аналіз виконання ТЗ, висновки	18.11.2024 – 24.11.2024	
10	Оформлення пояснювальної записки	25.11.2024 – 30.11.2024	
11	Попередній захист та доопрацювання МКР	28.11.2024 – 01.12.2024	
12	Перевірка магістерської роботи на наявність плагіату	02.12.2024 – 14.12.2024	
13	Представлення МКР до захисту	13.12.2024 – 16.12.2024	
14	Захист МКР	17.12.2024 – 20.12.2024	

Студент _____ Данило ГОНЧАР

Керівник роботи _____ Віталій ЛУКІЧОВ

АНОТАЦІЯ

УДК 004.056.5

Данило ГОНЧАР. Метод та засіб аналізу мережевих подій для SIEM систем. Магістерська кваліфікаційна робота зі спеціальності 125 – Кібербезпека та захист інформації, освітня програма – Безпека інформаційних і комунікаційних систем. Вінниця: ВНТУ, 2024. 84 с.

Українською мовою. Бібліографій: 25 назв; рис.: 27; табл. 12.

Магістерська кваліфікаційна робота присвячена покращенню процесу аналізу мережевих подій для SIEM систем шляхом розробки методу та засобу для цих цілей. Підготовлено технічно-економічне обгрунтування доцільності розробки. Здійснено аналіз принципів роботи систем аналізу мережевих подій, проведено порівняння існуючих засобів. Аналогічно здійснено порівняння існуючих технологій машинного навчання. Розроблено метод аналізу на основі використання гібридного підходу машинного навчання та розроблено засіб у вигляді програмного застосунку. В економічному розділі було проведено розрахунок витрат на розробку та можливого прибутку за залучення інвесторів.

Ілюстративна частина складається з плакатів з демонстрацією схем алгоритмів роботи системи та її тестуванням.

Ключові слова: SIEM система, машинне навчання, мережева подія, кластеризація, класифікація, системні логи.

ABSTRACT

UDC 004.056.5

Danylo HONCHAR. Method and Tool for Analyzing Network Events for SIEM Systems. Master's Thesis in the Specialty 125 – Cybersecurity, Educational Program – Security of Information and Communication Systems. Vinnytsia: VNTU, 2024. 84 pages.

Language: Ukrainian. Bibliography: 25 references; Figures: 27; Tables: 12.

This master's thesis is dedicated to improving the process of analyzing network events for SIEM systems by developing a method and a tool for this purpose. A technical and economic feasibility study of the proposed solution was prepared. An analysis of the operating principles of network event analysis systems was conducted, along with a comparison of existing tools. A similar comparison of existing machine learning technologies was also performed. A method for analysis based on a hybrid machine learning approach was developed, and a tool was implemented as a software application. The economic section includes a calculation of development costs and potential profits when attracting investors.

The illustrative part consists of posters demonstrating the system’s workflow algorithms and its testing process.

Keywords: SIEM system, machine learning, network event, clustering, classification, system logs.

ЗМІСТ

ВСТУП

У сучасному світі інформаційні системи відіграють ключову роль у функціонуванні бізнесу, державних установ та суспільства загалом. З розвитком цифрових технологій значно зростає кількість кібератак, що спричиняє суттєві загрози для конфіденційності, цілісності та доступності інформації. Однак, стандартні системи кіберзахисту часто не в змозі ефективно виявляти та реагувати на нові, складні загрози. Це підкреслює необхідність удосконалення методів аналізу мережевих подій з використанням сучасних підходів, таких як машинне навчання.

Предметом дослідження є метод та засіб аналізу мережевих подій в SIEM системах. Цей метод включає застосування алгоритмів машинного навчання, таких як кластеризація, виявлення аномалій та адаптивна класифікація, для автоматизованого ідентифікування підозрілих активностей, які можуть свідчити про кіберзагрози, зломи, витоки даних або інші мережеві інциденти.

Метою дослідження є покращення методу аналізу мережевих подій в SIEM системах шляхом застосування гібридного підходу машинного навчання, що поєднує кластеризацію, алгоритми виявлення аномалій та адаптивні методи класифікації для підвищення точності та швидкості ідентифікації потенційних загроз у мережевому трафіку.

Практична цінність полягає у створенні інструменту для підвищення ефективності роботи SIEM систем. Запропонована модель машинного навчання може бути інтегровані в існуючі SIEM платформи, що дозволить зменшити кількість хибних тривог, підвищити точність виявлення загроз, а також забезпечити динамічну адаптацію до нових типів атак.

Методами досліджень, використаними магістерській роботі, є: об’єктно-орієнтовані методи програмування, що є базовими для розробки програмного додатку.

Новизна роботи полягає в створенні інструменту що використовує гібридний підхід до машинного навчання. Гібридний підхід означає використання керованого та не керованого навчання залежно від даних які будуть використовуватись для аналізу. Розроблений метод і засіб можуть бути інтегровані в SIEM систему для подальшого використання експертами. Це дозволить суттєво покращити процес аналізу мережевих подій та буде сприяти зниженню ризиків мережевої безпеки як на персональних комп’ютерах, так і в великих корпоративних мережах.

Апробація була проведена на конференції Молодь в науці у вигляді 2 тез. Посилання на тези наведені в списку використаних джерел.

1 АНАЛІЗ ІНФОРМАЦІЙНИХ ДЖЕРЕЛ

1.1Аналіз проблеми захисту інформації

У сучасному світі цифрових технологій захист інформації стає однією з найбільш актуальних і важливих проблем. Зростання обсягу та складності кіберзагроз вимагає від організацій застосування нових підходів і рішень для забезпечення безпеки своїх інформаційних ресурсів. Щороку кількість кібератак продовжує збільшуватися, стаючи все більш витонченими та важко передбачуваними. Кіберзлочинці використовують різноманітні методи проникнення та маніпуляції даними, що створює серйозні ризики для інформаційних систем і критичної інфраструктури.

Однією з ключових проблем, з якими стикаються організації, є неефективність традиційних засобів захисту інформації [1]. Системи, побудовані на основі сигнатур або статичних правил, часто виявляються неспроможними ефективно виявляти нові, раніше невідомі загрози. Це призводить до великої кількості хибних тривог і недостатнього виявлення складних атак, таких як атаки нульового дня або багатовекторні загрози, що використовують раніше невідомі вразливості. Крім того, сучасні технології дозволяють кіберзлочинцям обходити традиційні засоби захисту, що робить їх менш ефективними в умовах динамічно змінюваних загроз.

Іншою важливою проблемою є обробка великих обсягів даних, що генеруються сучасними мережами. Щодня в інформаційних системах відбувається величезна кількість подій і транзакцій, які необхідно ретельно аналізувати для своєчасного виявлення можливих загроз. Проте, аналіз такої кількості даних стає дедалі складнішим і вимагає значних обчислювальних ресурсів, а також застосування новітніх методів для обробки та аналізу. Не менш важливою є необхідність автоматизації процесу аналізу подій, оскільки людські ресурси часто виявляються обмеженими або недостатніми для обробки величезних масивів даних в реальному часі.

У свою чергу, автоматизація процесів і впровадження інноваційних технологій стикаються з проблемою адаптації до існуючої інфраструктури. Багато організацій вже мають розгорнуті системи безпеки, і інтеграція нових рішень може вимагати значних зусиль та ресурсів. Проте з огляду на необхідність швидкої реакції на нові типи загроз і забезпечення адекватного рівня захисту, пошук ефективних та масштабованих рішень є нагальною потребою для багатьох підприємств.

Таким чином, основна проблема захисту інформації сьогодні полягає в невідповідності традиційних методів новим викликам. З розвитком кіберзагроз виникає потреба в нових підходах до захисту інформаційних систем, які здатні реагувати на загрози в реальному часі та забезпечувати надійний захист критичних даних.

1.2 Принципи аналізу мережевих подій

Принципи аналізу мережевих подій базуються на моніторингу та інтерпретації активності в мережі для виявлення аномалій, потенційних загроз і порушень безпеки. Це важливий процес для виявлення кібератак на ранніх етапах та їх подальшого запобігання [2]. Мережеві події включають взаємодії між різними пристроями, з'єднання, передачу даних, спроби доступу до ресурсів та інші дії, які відбуваються в комп'ютерній мережі.

Процес аналізу мережевих подій починається зі збору даних. Джерелами подій можуть бути:

- мережеві пристрої (маршрутизатори, комутатори, брандмауери), які записують логи про пересування трафіку в мережі;
- хости та сервери, які генерують логи про активність користувачів, використання ресурсів, помилки, спроби входу в систему тощо;
- системи безпеки (IPS/IDS, антивіруси), що виявляють і записують спроби вторгнень;
- засоби моніторингу трафіку (Packet sniffers, NetFlow), які записують потоки даних, що передаються мережею;

— журнали операційних систем та програмного забезпечення.

Мета збору даних полягає в накопиченні максимальної кількості інформації, що відображає стан та діяльність мережевої інфраструктури.

Зібрані події можуть мати різні формати, тому наступним кроком є нормалізація та уніфікація цих даних. Це процес перетворення даних до стандартизованого формату для того, щоб їх можна було ефективно аналізувати. Нормалізація включає:

1) виділення ключових атрибутів події (час події, IP-адреса джерела та призначення, тип події);
2) перетворення даних з різних форматів у єдиний вигляд для полегшення подальшого аналізу;
3) видалення зайвої або дубльованої інформації для зменшення обсягу даних.

Агрегація допомагає скоротити кількість оброблюваних подій і зосередитися на тих, що мають найбільшу значимість для безпеки мережі. Мережеві системи генерують величезні обсяги даних, які можуть містити повторювані або схожі події. Агрегація дозволяє об'єднувати ці події на основі спільних характеристик, наприклад, однакові IP-адреси, порти, типи подій тощо. Наприклад, тисячі запитів до одного веб-сервера за короткий проміжок часу можна об'єднати в одну подію з відповідною кількістю повторень.

Вона також допомагає структурувати події, зосереджуючи увагу на значущих групах дій. Наприклад, всі події, пов’язані з певним типом атаки, можуть бути зібрані в один запис, що спрощує розуміння загальної картини атаки. Часто події, які трапляються в один і той же часовий проміжок, можуть бути об'єднані в одну подію з часовими мітками, що відображають загальну активність у певний період часу. Це корисно для виявлення повторюваних дій або масованих атак за короткий проміжок часу.

У процесі агрегації виокремлюються ключові атрибути подій, наприклад, джерело або мета атак, типи запитів або відповідей, що дозволяє швидше визначати джерело загрози та її можливий вплив. Вона допомагає зменшити

навантаження на систему оскільки дозволяє аналізувати узагальнені події, а не обробляти кожну окремо. Це робить процес моніторингу мережі більш ефективним і швидким. Наприклад можна проводити зведення подій, пов'язаних зі скануванням портів, в один запис з деталізацією атакуючих IP і часу виконання сканування, або об'єднувати численні запитів з однієї IP-адреси в одну подію із зазначенням кількості запитів.

Також одним з важливих принципів аналізу мережевих подій є фільтрація цих подій. Це процес відсіювання подій, які не є значущими з точки зору кібербезпеки або не потребують подальшого аналізу [3]. Фільтрація зосереджує увагу на потенційно небезпечних або підозрілих подіях, що значно полегшує роботу системи.

Так наприклад важливо проводити відсівання шуму, тому що у мережах щоденно генерується величезна кількість подій, більшість з яких є рутинними або не несуть загроз. Це можуть бути системні логи, події, пов'язані з нормальним функціонуванням програм, або внутрішня активність мережевих пристроїв. Фільтрація дозволяє позбутися цих «шумових» подій, зосередившись на тих, що мають реальний потенціал загрози.

Для роботи систем аналізу мережевих подій дуже важливо проводити фільтрацію за рівнем важливості, так як ці події мають різні рівні — від інформаційних (інформативні події, що не несуть загроз) до критичних (події, що свідчать про можливий злам системи або серйозну атаку). Фільтрація дозволяє відсівати інформаційні події і зосереджуватися на тривожних сигналах. Фільтрація допомагає мінімізувати кількість подій, що потребують уваги, і зосередитися на більш суттєвих питаннях безпеки. Це забезпечує ефективність роботи систем аналізу та прискорює реагування на можливі загрози.

Агрегація та фільтрація подій працюють спільно, забезпечуючи ефективний аналіз великих обсягів мережевих даних. Агрегація допомагає зменшити кількість подій, об'єднуючи їх за ключовими параметрами, тоді як фільтрація дозволяє зосередитися на найбільш суттєвих або підозрілих подіях, відсіюючи неважливі або рутинні. Це робить аналіз мережевих подій більш керованим та ефективним.

Після цих етапів відбувається кореляція подій - процес встановлення взаємозв'язку між різними подіями, які можуть на перший погляд здаватися несуміжними, але разом можуть свідчити про певний шаблон або поведінку, характерну для загрози безпеці [4]. Основна мета кореляції — виявити складні та багатоступеневі атаки, де окремі події не здаються небезпечними, але в сукупності вони можуть створити серйозну загрозу.

Основні принципи кореляції:

—зв'язування подій за спільними ознаками: Це може включати час події, IP-адреси, імена користувачів або інші параметри. Наприклад, спроби входу в систему з різних IP-адрес за короткий період можуть бути пов'язані та свідчити про атаку методом «брутфорс»;

—пошук шаблонів: Кореляція дозволяє виявляти шаблони, які можуть сигналізувати про наявність атаки. Наприклад, якщо спостерігається послідовність подій, таких як невдалі спроби входу в систему, зміна прав користувача, а потім масове скачування даних, це може свідчити про успішну компрометацію облікового запису;

—використання правил кореляції: Багато систем використовують попередньо налаштовані правила кореляції. Наприклад, правило може свідчити, що якщо на сервері спостерігаються кілька несанкціонованих спроб входу і одночасно з'являються спроби сканування портів з того ж IP, це підвищує ймовірність того, що це спланована атака;

—часові рамки: Події, що відбуваються одночасно або в межах певного часового інтервалу, можуть бути взаємопов'язані. Наприклад, якщо після кількох спроб доступу до сервера з’являється несподіване відключення або зміна налаштувань, це може бути ознакою атаки;

—виявлення багатоступеневих атак: Наприклад, хакер може спочатку сканувати мережу для виявлення вразливих систем, потім спробувати отримати доступ і нарешті провести витік даних. Окремо ці події можуть виглядати нешкідливими, але їхній зв'язок свідчить про цільову атаку.

Кореляція подій дозволяє побачити повну картину атаки і зрозуміти її складові. Це критично важливо для виявлення складних загроз, які можуть складатися з кількох малопомітних подій.

Після кореляції буде відбуватись вже сам аналіз аномалій. Аналіз аномалій це процес виявлення подій або поведінки, які відхиляються від нормальних, очікуваних або попередньо визначених патернів мережевої активності [5]. Він ґрунтується на порівнянні поточної активності з історичними даними або встановленими нормами, щоб виявити підозрілі відхилення, які можуть вказувати на загрозу.

Він може мати такі складові:

1) визначення нормальної поведінки: Починається з визначення того, що вважається «нормальною» активністю в мережі або системі. Це може включати типові мережеві запити, обсяги трафіку, частоту доступу до певних ресурсів тощо. На основі цих норм будується базовий рівень (baseline), з яким потім порівнюються всі події;
2) виявлення відхилень: Будь-яка подія, яка суттєво відрізняється від нормальних показників, розглядається як потенційна аномалія. Наприклад, різке збільшення обсягу завантаженого трафіку, численні спроби авторизації протягом короткого часу, або доступ до систем з невідомих IP-адрес можуть бути ознаками аномальної поведінки;
3) адаптивне навчання: Системи, що використовують аналіз аномалій, можуть використовувати методи машинного навчання для адаптації до змін у поведінці мережі. Це означає, що з часом система може коригувати свій базовий рівень, вивчаючи нові патерни нормальної поведінки і виявляючи аномалії, навіть коли вони змінюютьс;
4) контекстуальний аналіз: Важливим аспектом аналізу аномалій є здатність системи розуміти контекст події. Наприклад, високий рівень трафіку може бути нормальним під час робочих годин, але той самий рівень трафіку вночі може вважатися аномалією;

5) аналіз поведінки користувачів і пристроїв: Аналіз аномалій також може застосовуватися для виявлення відхилень у поведінці користувачів або пристроїв. Якщо користувач раптом починає взаємодіяти з файлами або системами, до яких раніше не мав доступу, або пристрій починає генерувати нетипові запити, це може бути ознакою компрометації.

Для аналізу використовується багато методів, але деякі є основними і найбільш поширеними, а саме:

—збір і аналіз історичних даних для створення базової лінії нормальної активності. Відхилення від цієї базової лінії можуть свідчити про аномалії;

—використання статистичних моделей для виявлення рідкісних подій, що не відповідають загальній статистичній нормі, наприклад, обчислення середніх значень, стандартного відхилення або процентильних діапазонів;

—використання алгоритмів, таких як кластеризація або нейронні мережі, для аналізу великих обсягів даних і виявлення прихованих аномалій на основі змін у поведінці мережі або користувачів.

Аналіз аномалій може виявляти нові загрози, які ще не були зафіксовані або зареєстровані в базах відомих сигнатур. Це робить його ефективним інструментом для боротьби з новими або складними загрозами [6]. Однак також можливі численні хибні спрацьовування, коли нормальна діяльність може бути розпізнана як аномалія. Тому важливо правильно налаштовувати базові лінії та застосовувати додаткові методи для мінімізації таких спрацьовувань.

Кореляція подій і аналіз аномалій доповнюють один одного. Кореляція допомагає виявляти взаємопов'язані події, створюючи загальну картину атаки, тоді як аналіз аномалій фокусується на виявленні нетипової поведінки, що може сигналізувати про нові або складні атаки. Разом ці методи підвищують ефективність систем виявлення загроз і швидше реагують на можливі інциденти.

Після виявлення аномалій або кореляції подій відбувається ідентифікація загроз. Цей етап включає зіставлення виявлених подій з відомими сценаріями атак або поведінковими шаблонами. Наприклад, після кореляції може бути визначено, що зафіксовані дії є класичною DDoS-атакою або спробою фішингового втручання. Виявлені загрози можуть класифікуватися за ступенем небезпеки та пріоритетом реагування.

Якщо виявлено загрозу, система надсилає повідомлення або попередження адміністраторам безпеки для подальших дій. Оповіщення може містити детальну інформацію про загрозу, включаючи джерело, тип події та рекомендовані дії. У деяких випадках можлива автоматична відповідь на загрозу, наприклад, блокування підозрілого трафіку або ізоляція скомпрометованого пристрою від мережі.

1.3Аналіз роботи SIEM-систем

В сучасному світі інформаційна безпека стала одним із ключових факторів для успішного функціонування організацій будь-якого масштабу. Кількість та складність кіберзагроз постійно зростає, що вимагає впровадження все більш ефективних і комплексних засобів захисту інформаційних систем. Одним із таких рішень є системи інформаційної безпеки та управління подіями (SIEM – Security Information and Event Management), які відіграють центральну роль у забезпеченні кібербезпеки, об’єднуючи збір, аналіз і моніторинг подій в режимі реального часу [7].

SIEM системи дозволяють організаціям отримувати повне уявлення про стан мережі та системи в будь-який момент часу, відслідковуючи підозрілі дії та інциденти, що можуть свідчити про можливі загрози. Використання SIEM стало особливо актуальним у зв'язку з розвитком складних багатоетапних атак, коли звичайний моніторинг окремих подій вже не може забезпечити належний рівень безпеки. Вони збирають величезні обсяги даних з різних джерел: файрволів, антивірусів, систем виявлення вторгнень, серверів, мережевих пристроїв, додатків тощо, і на основі цього здійснюють комплексний аналіз для виявлення потенційних загроз.

Основна ідея функціонування SIEM полягає в об'єднанні подій, що можуть бути незначними або непомітними самі по собі, в одну картину, що дозволяє виявляти складні загрози на ранніх етапах [8]. Це досягається через застосування таких технологій, як кореляція подій, аналіз аномалій, фільтрація, агрегація та інші методи обробки даних, що дозволяють підвищити ефективність виявлення загроз.

Враховуючи сучасні виклики у сфері інформаційної безпеки, аналіз роботи SIEM систем стає важливим інструментом для оцінки їх ефективності та розуміння, як саме вони допомагають організаціям захищати свої ресурси. Важливо розуміти, як відбувається збір, обробка і аналіз даних, які механізми використовуються для виявлення загроз, і які підходи можуть бути впроваджені для покращення цих процесів.

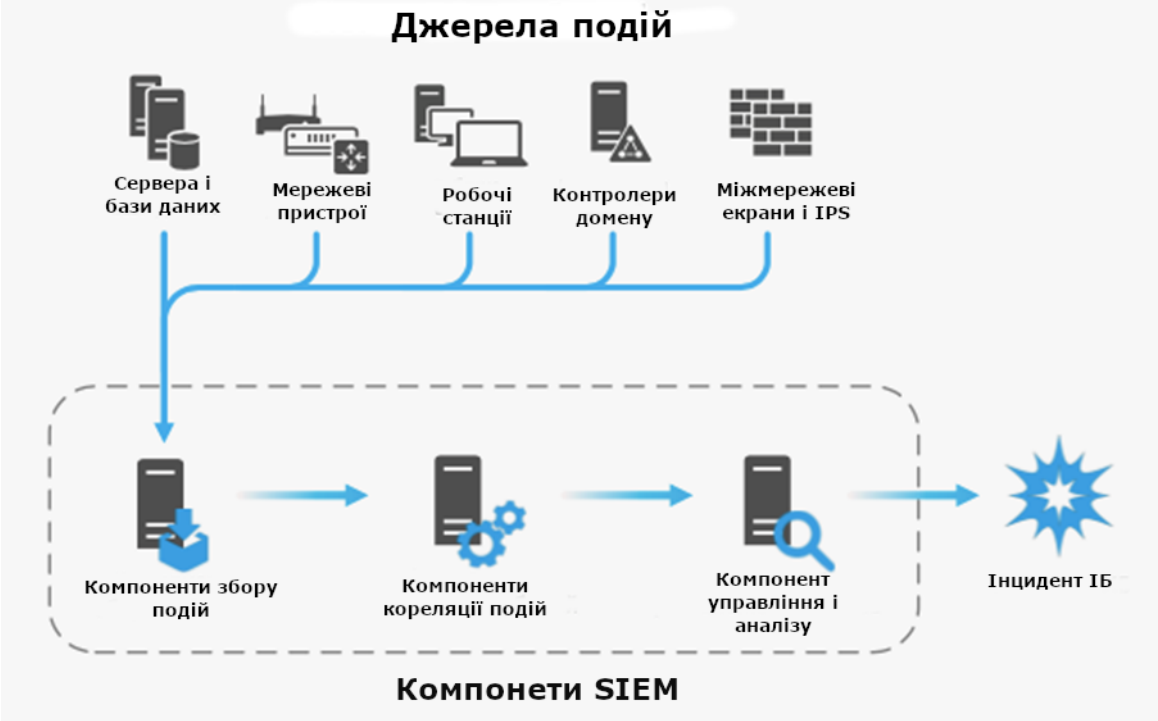


Рисунок 1.1 – схема роботи SIEM систем

SIEM системи поєднують кілька функціональних елементів, кожен з яких грає важливу роль у загальному процесі захисту інформаційної інфраструктури. Нижче розглянемо більш детально, як саме працюють основні механізми SIEM, зважаючи на раніше описані фактори.

Обсяги даних можуть бути надзвичайно великими, що створює серйозні виклики для системи в контексті обробки та зберігання цієї інформації. SIEM

системи мають забезпечувати безперервний збір даних у реальному часі, що дозволяє отримувати актуальну інформацію про стан мережевої інфраструктури і швидко реагувати на будь-які відхилення [9].

Однак основна проблема на цьому етапі полягає у різноманітності форматів та структур даних, що надходять з різних джерел. Це робить необхідною процедуру нормалізації, яка є другим важливим етапом роботи SIEM систем.

Нормалізація дозволяє перетворити дані з різних джерел у єдиний формат, що значно спрощує їх подальший аналіз. Наприклад, логи подій з файрвола можуть мати одну структуру, тоді як логи з серверів або додатків – зовсім іншу. SIEM системи приводять ці дані до спільного формату, забезпечуючи можливість їх порівняння і ефективної обробки.

Це особливо важливо в контексті кореляції подій, оскільки без нормалізації дані з різних джерел могли б бути важко інтерпретовані для виявлення потенційних загроз. Завдяки нормалізації SIEM системи можуть будувати зв'язки між подіями та знаходити закономірності, які свідчать про наявність загроз.

Через велику кількість подій, що генеруються щодня, SIEM системи мають можливість агрегації подій для зменшення обсягу даних, які підлягають аналізу [10]. Агрегація дозволяє групувати схожі події або події, що повторюються, в одну категорію. Наприклад, якщо від одного користувача за короткий проміжок часу надійшло кілька запитів на доступ до системи, SIEM система може об'єднати ці події в одну, щоб уникнути дублювання інформації.

SIEM система за допомогою кореляції може поєднати різні події і визначити, що це є частиною одного інциденту безпеки. Це дозволяє виявляти більш складні атаки, такі як багатоетапні або "повільні" атаки, які можуть залишатися непоміченими при традиційному підході до аналізу подій.

Крім кореляції подій, SIEM системи також використовують аналіз аномалій, який дозволяє виявляти події, що виходять за межі звичайної поведінки системи. Аналіз аномалій зазвичай ґрунтується на статистичних

моделях, профілях поведінки або машинному навчанні, що дозволяє системі "вчитися" на основі попередніх подій і адаптуватися до змін у середовищі.

Наприклад, якщо зазвичай користувач входить в систему з однієї геолокації, але раптом спроба входу відбувається з іншого континенту, це може бути сигналом для SIEM системи про аномалію, яка потребує подальшого аналізу. Такі аномалії можуть вказувати на потенційні атаки або компрометацію облікових даних.

Аналіз аномалій особливо важливий для виявлення невідомих загроз, які можуть бути не відомі традиційним правилам і сигнатурам. SIEM системи здатні використовувати поведінковий аналіз для виявлення атак, що не відповідають жодному з відомих патернів.

Після того як SIEM система виявляє потенційну загрозу, вона ініціює процес оповіщення та реагування [11]. Це може включати автоматичну блокаду загроз, зміну політики безпеки або ізоляцію певних елементів мережі для запобігання подальшому поширенню атаки. Завдяки інтеграції з іншими системами безпеки SIEM забезпечує можливість оперативної реакції на інциденти, що є ключовим для зниження ризиків.

Таким чином, робота SIEM систем будується на багаторівневому підході до збору та аналізу подій, що дозволяє підвищити ефективність виявлення загроз і забезпечити високий рівень кібербезпеки в організаціях. Ключові механізми, такі як агрегація, фільтрація, кореляція та аналіз аномалій, роблять SIEM системи одним з найефективніших засобів захисту інформаційних ресурсів у сучасних умовах.

Для кращого розуміння які існуючі SIEM системи є максимально корисними для певного підприємства потрібно провести порівняння популярних рішень від різних компаній.

SIEM системи можна поділити на комерційні та безкоштовні. Комерційні рішення (Splunk, IBM QRadar, ArcSight) є потужними інструментами з широким функціоналом, проте вони коштують дорого, що може бути проблемою для невеликих організацій. Тому в даному аналізі пріоритет буде віддано

безкоштовним SIEM рішенням, таким як Wazuh, Graylog, ELK Stack, та Security Onion [12].

Wazuh – це сучасна SIEM система з відкритим вихідним кодом, яка поєднує в собі засоби моніторингу безпеки, аналізу логів та управління інцидентами.

Переваги:

—відкритий вихідний код – безкоштовне використання і можливість адаптації під потреби компанії;

—інтеграція з ElasticSearch – дозволяє швидкий пошук і аналіз логів.

—підтримка різних платформ (Windows, Linux, macOS);

—функціонал HIDS (Host Intrusion Detection System) для контролю змін у файлах;

—постійне оновлення правил для виявлення загроз (на основі MITRE ATT&CK).

Недоліки:

—висока ресурсоємність при обробці великої кількості подій;

—складність налаштування для великих мережевих інфраструктур;

—лімітації у масштабованості порівняно з дорогими комерційними SIEM.

Graylog – це SIEM рішення з відкритим вихідним кодом, яке спеціалізується на централізованому зборі та аналізі логів у реальному часі.

Переваги:

—зручний і гнучкий інтерфейс для аналізу логів;

—підтримує агрегацію подій з багатьох джерел та їх фільтрацію;

—можливість кореляції подій за допомогою додаткових правил;

—плагінна система дозволяє розширювати функціонал;

—гарно масштабується для середніх і великих мереж.

Недоліки:

—обмежений функціонал виявлення аномалій без інтеграції з іншими інструментами ML;

—необхідність значних налаштувань і технічних знань для ефективної роботи;

—при високому навантаженні можливе зниження продуктивності.

ELK Stack (Elasticsearch, Logstash, Kibana) – це набір інструментів з відкритим кодом, який широко використовується для збору, обробки та візуалізації даних, включаючи логи безпеки. Хоча сам ELK не є SIEM у традиційному розумінні, його часто використовують як основу для побудови SIEM систем.

Переваги:

—висока масштабованість та гнучкість;

—сильний модуль пошуку та аналізу даних через Elasticsearch;

—Logstash дозволяє обробляти та нормалізувати дані з різних джерел;

—Kibana надає потужні інструменти для візуалізації та моніторингу;

—інтеграція з різними модулями машинного навчання.

Недоліки:

—висока складність налаштування та адміністрування;

—високі вимоги до ресурсів для обробки великих обсягів даних;

—потребує додаткових налаштувань для кореляції та виявлення аномалій.

Security Onion – це комплексне рішення для моніторингу безпеки та аналізу подій з відкритим кодом, що включає ряд популярних інструментів (Suricata, Zeek, ELK).

Переваги:

—інтеграція з Suricata та Zeek для глибокого аналізу мережевого трафіку;

—повний набір інструментів для виявлення загроз та реагування;

—автоматизоване розгортання з передвстановленими конфігураціями;

—підтримує мережевий моніторинг та аналіз логів у реальному часі.

Недоліки:

—велика кількість компонентів ускладнює керування та оновлення системи;

—необхідність значних ресурсів для стабільної роботи у великих мережах;

—висока крива навчання через комплексність системи.

Якщо проводити узагальнення всіх описаних вище систем то їх можна описати через наступну таблицю

Таблиця 1.1 – порівняння популярних SIEM систем

SIEM Система	**Переваги**	**Недоліки**	**Цільове використання**
Wazuh	Відкритий код, інтеграція з ElasticSearch	Складність налаштування, ресурсомісткість	Моніторинг серверів та хостів
Graylog	Гнучкий інтерфейс, плагіни	Обмежений функціонал для аномалій	Аналіз логів у реальному часі
ELK Stack	Масштабованість, потужна візуалізація	Складне налаштування, великі вимоги	Інтеграція з ML, великі мережі
Security Onion	Інтеграція з Suricata, готові інструменти	Висока крива навчання, складне управління	Глибокий аналіз трафіку та логів

Завдяки відкритому коду, безкоштовні SIEM системи дозволяють розробляти унікальні рішення, інтегрувати їх з ML та адаптувати до конкретних потреб організації, що може бути основою для нових підходів у дослідженні.

1.4 Аналіз технологій машинного навчання

У сучасних умовах розвитку інформаційних технологій та зростання складності кіберзагроз, використання машинного навчання (ML) в системах інформаційної безпеки та управління подіями (SIEM) стало невід’ємною

частиною процесу виявлення, аналізу та реагування на загрози. Машинне навчання дозволяє автоматизувати багато аспектів аналізу мережевих подій, підвищуючи ефективність роботи SIEM систем завдяки здатності обробляти великі обсяги даних, виявляти аномалії та будувати прогнози на основі історичних даних [13].

Машинне навчання включає кілька основних підходів, які можна використовувати як в SIEM системах так і в будь якому іншому засобі:

- класифікація дозволяє відносити події до певних класів (наприклад, нормальні чи підозрілі). Використовуючи алгоритми класифікації, такі як дерева рішень, методи опорних векторів (SVM) або нейронні мережі, SIEM системи можуть автоматично визначати, які події є загрозами, а які – ні;
- за допомогою регресійних моделей SIEM системи можуть прогнозувати кількість загроз або аномальних подій на основі попередніх даних. Це дозволяє організаціям підготуватися до потенційних атак і вжити превентивних заходів;
- аналіз аномалій використовує статистичні підходи для виявлення відхилень від нормальної поведінки. Застосування алгоритмів, таких як методи кластеризації або ізоляційні ліси, дозволяє виявляти нові типи атак, які раніше не були зафіксовані;
- навчання з підкріпленням може бути використано для оптимізації реакції на загрози. Система отримує винагороду або покарання за свої дії в залежності від їх результативності, що дозволяє їй покращувати стратегії реагування на інциденти [14].

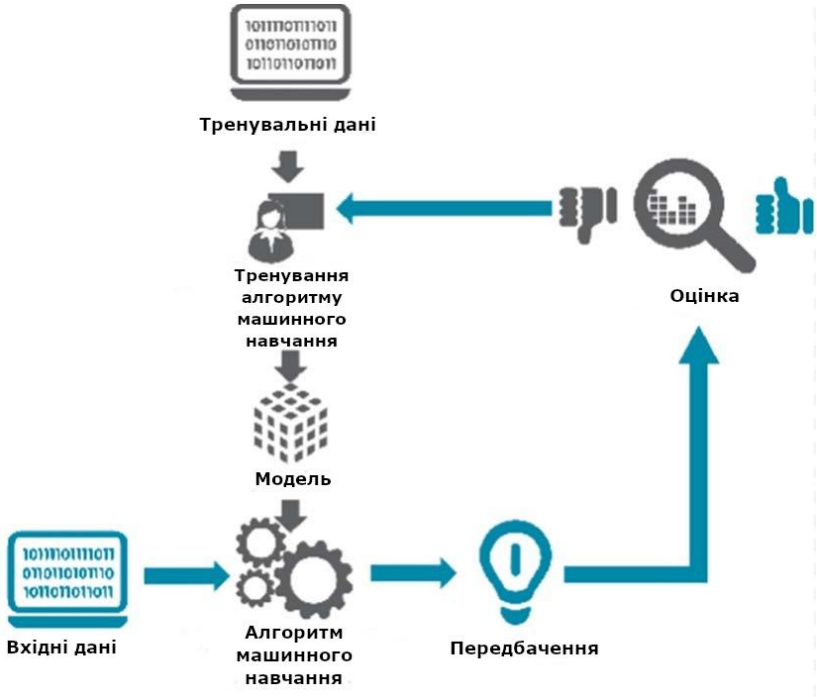


Рисунок 1.2 – Підхід машинного навчання

Машинне навчання забезпечує SIEM системи можливістю виявлення складних загроз, які можуть бути непомітними при традиційних методах моніторингу. Використовуючи алгоритми аналізу аномалій, SIEM може виявляти неочікувані або підозрілі дії, такі як спроби входу з нових геолокацій, незвичні патерни доступу до даних або аномальні обсяги трафіку. Це дозволяє швидше реагувати на потенційні загрози.

Машинне навчання також може бути застосоване для кореляції подій, де алгоритми можуть автоматично виявляти зв’язки між різними подіями в реальному часі. Це дозволяє SIEM системам більш ефективно аналізувати інформацію, знаходити складні закономірності і виявляти багатоетапні атаки.

Завдяки використанню машинного навчання, SIEM системи можуть автоматизувати процеси реагування на інциденти. Алгоритми можуть навчатися на основі попередніх інцидентів, щоб визначити найефективніші дії у відповідь на конкретні загрози. Це дозволяє скоротити час реагування і підвищити ефективність захисту [15].

Його також може бути використано для створення профілів поведінки користувачів та пристроїв у мережі. Це дозволяє SIEM системам розуміти, що є нормальним для кожного користувача або системи, а також виявляти відхилення від цього профілю, які можуть свідчити про кіберзагрози.

Впровадження машинного навчання в SIEM системи не обходиться без викликів. По-перше, потрібен великий обсяг якісних даних для навчання моделей, що може бути проблемою для деяких організацій. По-друге, моделі машинного навчання можуть потребувати постійної адаптації до нових загроз, оскільки кіберзлочинці постійно вдосконалюють свої методи [16].

Проте, перспективи використання машинного навчання в SIEM системах виглядають багатообіцяючими. Постійний розвиток технологій та алгоритмів дозволяє створювати все більш потужні рішення для виявлення та реагування на кіберзагрози. Завдяки впровадженню машинного навчання, SIEM системи можуть стати більш ефективними, адаптивними і здатними реагувати на нові виклики в сфері інформаційної безпеки.

Для аналізу мережевих подій і підвищення ефективності SIEM систем можна застосовувати кілька популярних підходів машинного навчання. У цій частині розглянемо, які саме технології підходять для виявлення аномалій**,** класифікації загроз**,** кореляції подій **та** прогнозування інцидентів [17].

Кластеризація використовується для групування подій без попереднього знання про класи (навчання без вчителя). Цей метод дозволяє виявляти аномалії та нові види загроз.

Приклади алгоритмів:

—K-Means;

—DBSCAN (Density-Based Spatial Clustering);

—ізоляційний ліс (Isolation Forest).

Переваги:

—не потребує попередньо маркованих даних;

—ефективний для виявлення аномальних патернів у великих масивах даних;

—ізоляційний ліс добре працює для детектування рідкісних подій.

Недоліки:

—може давати помилкові позитивні результати при надмірній кількості нормальних подій;

—алгоритми, як-от K-Means, можуть мати складність із вибором оптимальної кількості кластерів [18].

Приклад використання:

Ізоляційний ліс у SIEM системах може автоматично виявляти незвичайні мережеві активності, які не відповідають звичній поведінці користувачів або пристроїв.

Класифікаційні моделі застосовуються для ідентифікації загроз, розділяючи події на підозрілі та безпечні на основі навчання на маркованих даних .

Приклади алгоритмів:

—Random Forest;

—Support Vector Machine (SVM);

—логістична регресія;

—нейронні мережі.

Переваги:

—добре працюють для класифікації відомих загроз;

—Random Forest стійкий до надмірної підгонки (overfitting);

—нейронні мережі здатні виявляти складні патерни у великих даних.

Недоліки:

—потребують великого обсягу маркованих даних для навчання;

—SVM може бути малоефективним при великій кількості даних;

—нейронні мережі потребують значних обчислювальних ресурсів.

Приклад використання:

Random Forest може використовуватися для автоматичної класифікації подій, таких як спроби входу, на категорії «нормальні» та «підозрілі».

Аналіз аномалій дозволяє виявляти відхилення від нормальної поведінки в мережевому середовищі та знаходити нові види атак. Використовує статистичні та ML підходи для виявлення рідкісних або незвичних подій.

Приклади алгоритмів:

—Autoencoders;

—Isolation Forest;

—One-Class SVM.

Переваги:

—не потребує маркованих даних;

—може виявляти нові та невідомі загрози;

—Autoencoders здатні працювати з великою кількістю параметрів та складними даними.

Недоліки:

—висока ймовірність помилкових спрацьовувань у випадку різноманітних нормальних подій;

—Autoencoders можуть бути складними у налаштуванні та потребують значних ресурсів.

Приклад використання:

Autoencoder може використовуватися для моніторингу трафіку в реальному часі та виявлення нетипових патернів.

Цей підхід використовує середовище навчання, де агент отримує винагороду за правильні дії та покарання за помилкові. Він може бути використаний для оптимізації реагування на інциденти.

Приклади алгоритмів:

—Q-Learning;

—Deep Q-Networks (DQN).

Переваги:

—може адаптуватися до нових загроз та умов;

—використовується для автоматизації прийняття рішень в системах реагування на загрози.

Недоліки:

—потребує багато часу та даних для навчання;

—важко застосувати в умовах, коли немає чітких правил для винагороди та покарання.

Приклад використання:

Q-Learning може бути застосований для оптимізації стратегії блокування підозрілих IP-адрес в реальному часі.

NLP може використовуватися для аналізу журналів логів та повідомлень про інциденти, виявляючи ключові патерни та кореляції в текстових даних.

Приклади алгоритмів:

—Word2Vec;

—BERT (Bidirectional Encoder Representations).

Переваги:

—дозволяє автоматизувати аналіз текстових даних (логи, звіти про загрози);

—може знаходити зв’язки між різними подіями на основі текстової інформації.

Недоліки:

—потребує великих обсягів даних для навчання;

—складність у налаштуванні моделей для специфічних випадків.

Приклад використання:

BERT може використовуватися для аналізу логів та автоматичного виділення критичних подій.

Таблиця 1.2 – Порівняння технологій машинного навчання

Технологія	**Призначення**	**Переваги**	**Недоліки**	**Приклад використання**
Кластеризація	Виявлення аномалій	Не потребує маркованих даних	Помилкові позитивні спрацьовування	Ізоляція рідкісних подій
Класифікація	Ідентифікація загроз	Висока точність для відомих атак	Потребує маркованих даних	Розподіл подій на категорії
Аналіз аномалій	Виявлення нових загроз	Виявляє невідомі загрози	Висока кількість хибних тривог	Виявлення аномального трафіку
Навчання з підкріпленням	Оптимізація реакції на загрози	Адаптація до нових умов	Потребує часу на навчання	Автоматизація реагування
NLP	Аналіз текстових логів	Автоматизує обробку текстових даних	Складність налаштування	Виділення ключових подій з логів

Кожен із розглянутих підходів машинного навчання має свої переваги та недоліки. Для побудови ефективної SIEM системи рекомендується використовувати гібридні підходи, комбінуючи кілька технологій. Наприклад, кластеризацію можна поєднати з класифікацією для виявлення як відомих, так і невідомих загроз, а NLP можна застосовувати для автоматизації аналізу логів. Це дозволить створити більш адаптивну та ефективну систему для моніторингу мережевих подій і захисту від кібератак.

1.5 Постановка задачі

У сучасному цифровому середовищі, де кібербезпека є одним із ключових аспектів захисту даних та інформаційних систем, розробка ефективних методів аналізу мережевих подій набуває особливої актуальності. Кіберзловмисники активно використовують комп'ютерні технології для реалізації своїх протиправних цілей.

Їх дії можуть бути спрямовані на атаки комп'ютерних систем та мереж з метою отримання несанкціонованого доступу до даних, викрадення коштів або

нанесення шкоди. Зловмисники також використовують шкідливе програмне забезпечення для розповсюдження дезінформації, вимагання або здійснення інших незаконних операцій.

Потреба у створенні нового методу виявлення та моніторингу потенційних кіберзловмисників є важливим завданням. Реалізація такого підходу дозволить значно підвищити ефективність розслідування кіберінцидентів у мережевих середовищах та покращити загальний рівень кібербезпеки.Мета даної роботи полягає у створенні інноваційного підходу до виявлення та моніторингу аномальної активності в мережевих системах, що забезпечить високий рівень захисту інформаційних ресурсів.

Зважаючи на широкий спектр потенційних кіберзагроз, цей підхід зосереджений на аналізі подій, які можуть свідчити про дії зловмисників у мережах організацій або публічних сервісах. Це має особливе значення в умовах боротьби з кібератаками, запобігання розголошенню конфіденційної інформації, а також виявлення аномальної поведінки, яка може загрожувати безпеці критичних систем.

2. МЕТОД АНАЛІЗУ МЕРЕЖЕВИХ ПОДІЙ ДЛЯ SIEM-СИСТЕМИ

2.1 Метод аналізу мережевих подій

Машинне навчання використовується в багатьох методах аналізу мережевих подій для забезпечення швидшого й точного реагування на аномалії та потенційні загрози. Але як і в аналізі подій в машинному навчанні є безліч методів та принципів які можна використовувати для роботи. Одні методи можуть бути кращі за інші в точності, інші в швидкості, тому для забезпечення максимальної продуктивності засобу було запропоновано гібридний підхід. Гібридний підхід використовує поєднання алгоритмів керованого (Supervised) і некерованого (Unsupervised) навчання, щоб, з одного боку, виявляти вже відомі загрози, а з іншого знаходити аномальні патерни, що можуть вказувати на нові типи загроз.

Метод базується на алгоритмах кластеризації та виявлення аномалій, адаптованих для роботи з великими обсягами даних у реальному часі. Основною метою є покращення виявлення потенційних загроз і мінімізація хибних спрацювань шляхом аналізу аномальних подій та адаптації порогових значень на основі специфіки мережевого трафіку.

Кероване навчання в методиці базується на алгоритмах класифікації подій та логістичної регресії. Класифікація покращує точність за рахунок різного впливу кожного сусіда на прийняття рішення. Підхід корисний для виявлення аномалій або класифікації мережевих подій, коли деякі сусіди мають більшу вагу через близькість або інші характеристики. А логістична регресія оцінює ймовірність належності об’єкта до певного класу використовуючи логістичну функцію. Проте, у методиці вводиться адаптивна зміна ваг кожної ознаки, що дозволяє враховувати поточний стан мережі, особливості трафіку й інші фактори.

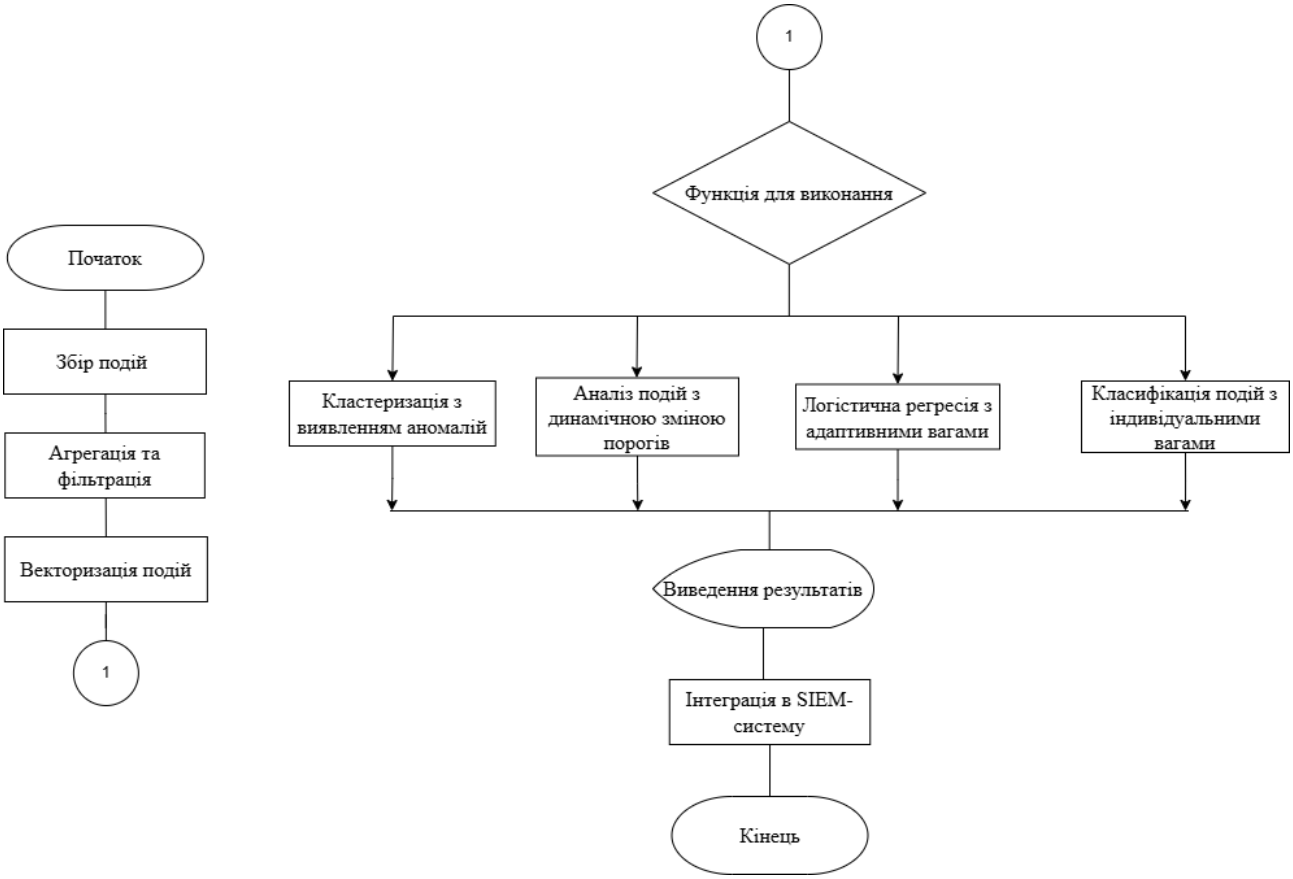


Рисунок 2.1 – Загальна схема роботи засобу

Для того щоб краще зрозуміти в чому суть поєднання декількох методів машинного навчання потрібно розглянути кожен з них окремо щоб зрозуміти за що вони відповідають та для чого використовуються в програмі.

K-Means з аналізом аномалій (K-Means Anomaly Detection) — це варіант стандартного K-Means, який не тільки виконує кластеризацію, але й виявляє аномальні події всередині кожного кластера. Такий підхід дозволяє відслідковувати не тільки групи подій, але й виявляти підозрілі або аномальні дії в цих групах.

1) алгоритм розбиває події на кластери, виходячи зі схожості атрибутів подій. Наприклад, події можуть групуватися за такими параметрами, як джерело, IP-адреса, тип запиту, час виникнення тощо. Центроїд обчислюється для кожного кластера, представляючи "середнє" положення подій, які належать до цього кластера;

2) після формування кластерів кожній точці (події) обчислюється відстань до її кластера (від центроїда). Це значення показує, наскільки подія відрізняється від типових подій у цьому кластері. Для вимірювання відстані використовується метрика Євклідова відстань, яка представляє з себе корінь суми всіх квадратів різниць між координатами точки та центроїда. $d = \sqrt{\sum_{i=1}^{n}(x_i - c_i)^2}$, де x_i – координата точки, c_i – координата центроїда, а n – кількість вимірів (факторів події, таких як IP-адреса, порт, часовий інтервал тощо);
3) для кожного кластера встановлюється порогове значення відстані, яке визначає межу "нормальності" для подій цього кластера. Поріг обирається на основі персентилю. Він встановлює поріг для кластерів, вище якого події вважаються аномаліями. Ідея полягає в тому, щоб визначити, наскільки відстань кожної події від центроїда наближається до «нормальних» значень у кластері і чи виходить вона за ці значення. Це дозволяє динамічно налаштовувати пороги в залежності від характеру даних. Наприклад, 90-й персентиль означає, що 90% значень відстаней до центроїда в кластері менші або рівні цьому значенню, а 10% — більші. Це дозволяє визначити поріг для рідкісних подій;
4) події, що перевищують встановлений поріг, позначаються як аномальні. Наприклад, якщо подія має набагато більшу відстань від центроїда свого кластера, це може свідчити про те, що подія є відмінною від основної групи подій і потребує додаткової уваги. Такі події можуть сигналізувати про потенційно небезпечні або підозрілі дії в мережі, зокрема, вторгнення або інші загрози;
5) після завершення аналізу система формує список аномальних подій. Кожна подія супроводжується інформацією про її відстань до центроїда та відповідний кластер. Ці події можуть бути направлені на подальший аналіз, а також використані для відправки сповіщень чи для детального розгляду співробітниками служби безпеки.

Рисунок 2.2 – Алгоритм роботи кластеризації з аналізом аномалій

Цей метод підходить для групування подій, пов’язаних із спробами входу в систему, веб-запитами або змінами у внутрішній мережі. Події, які суттєво відрізняються від основної маси за певними атрибутами (наприклад, географічне місце джерела, нетипова IP-адреса або час доступу), можуть бути виявлені як потенційні аномалії і направлені на подальшу перевірку. Система самостійно ідентифікує підозрілі події, що полегшує роботу співробітників безпеки та він може застосовуватися на великих наборах даних і легко адаптується до різних типів подій.

Ще одним методом для аналізу мережевих подій використовується Isolation Forest з динамічною зміною порогів на основі періодичності. Це підхід до адаптації SIEM системи з використанням методу Isolation Forest для мережевих подій, що характеризуються зміною інтенсивності в різні періоди. Багато SIEM-систем налаштовані на статичні пороги, що не враховує змінну природу мережевого трафіку. Використання динамічних порогів дозволяє підвищити точність виявлення аномалій шляхом автоматичного налаштування алгоритму під певний проміжок часу або умови. Динамічний поріг розраховується на основі історичних даних про мережевий трафік, збір яких допомагає визначити звичайні та пікові періоди. Це дозволяє побудувати змінний поріг, що враховує, наприклад, добові або тижневі коливання активності.

Для цього процес можна розділити на кілька кроків:

1) аналіз історичних даних для виділення періодів активності: Данні за попередні періоди аналізуються для виявлення закономірностей — періодів із підвищеною або зниженою активністю. Наприклад, за даними за тиждень можна виявити, що трафік зростає в певний час робочих днів і знижується вночі або на вихідних;
2) побудова періодичних інтервалів і динамічного порогу для кожного з них: Залежно від закономірностей можна виділити інтервали (робочий день, вихідний день, нічний час). Для кожного інтервалу розраховується статистика аномальних і нормальних подій. На основі середніх значень та дисперсії в кожному інтервалі розраховується динамічний поріг;
3) використання коефіцієнта для налаштування порогу: На базі історичних даних для кожного інтервалу розраховується коефіцієнт, який коригує поріг аномалій для Isolation Forest. Наприклад, якщо середня кількість подій у піковий період зростає на 30%, поріг аномалій також збільшується на цей відсоток;

4) адаптація порогу в реальному часі: У міру того, як нові дані надходять у систему, алгоритм автоматично коригує поріг відповідно до поточного періоду. Наприклад, коли починається робочий день, поріг підвищується, щоб уникнути помилкових спрацьовувань під час збільшення трафіку.

Наприклад, якщо середня кількість з'єднань за робочий день становить 500, а вночі знижується до 100, то відповідно Isolation Forest буде налаштований так, що пороговий рівень вночі буде нижчим. Таким чином, вночі система буде більш чутливою до аномалій, коли кожна подія відразу перевіряється на відхилення, тоді як вдень допускаються більш високі порогові значення для пом'якшення помилкових спрацьовувань.

Завдяки коригуванню порогу система менш схильна реагувати на події, які є нормальними для певного періоду, також чутливість до аномалій зростає в періоди низької активності (наприклад, вночі), що допомагає виявляти підозрілі події, які в робочий час можуть залишатися непоміченими.

Сам алгоритм буде виглядати наступним чином:

1) початковий етап, де дані про мережевий трафік збираються протягом певного часу (тиждень, місяць) для визначення патернів та закономірностей. Визначаються характерні періоди активності мережі (наприклад, "робочий день" і "нічний період" або "вихідний день" і "робочий день"). Історичні дані збираються окремо для кожного з цих періодів, щоб зрозуміти їх специфіку — середню активність, кількість запитів, пік трафіку та інші показники;
2) далі система обробляє історичні дані, виділяючи періоди підвищеної та зниженої активності. Для кожного з визначених періодів розраховується середнє значення активності та стандартне відхилення. Середнє значення відображає типову активність у цей період, а стандартне відхилення показує ступінь розсіювання даних (наскільки активність може відхилятися від середнього);

3) на основі середніх значень та стандартних відхилень для кожного періоду визначаються порогові значення. Для пікових періодів встановлюється вищий поріг, для менш активних періодів — нижчий;
4) розраховані пороги інтегруються в систему, яка тепер знає, що під час кожного періоду вона повинна застосовувати відповідне порогове значення для виявлення аномалій. Це дозволяє алгоритму виявляти аномалії, адаптуючи чутливість залежно від поточного періоду активності;
5) надходження нових даних у реальному часі, кожна подія перевіряється алгоритмом Isolation Forest із використанням порогового значення, встановленого для поточного періоду;
6) система постійно адаптує пороги на основі нових вхідних даних і поточного періоду, що дозволяє підвищити ефективність виявлення аномалій. На основі даних за останні кілька місяців або тижнів можна щоквартально перераховувати пороги, щоб відобразити зміни у характері активності.

Рисунок 2.3 – Алгоритм роботи аналізу мережевих подій з динамічною зміною порогів

Цей метод у SIEM-системі може аналізувати потоки даних у реальному часі, виділяючи потенційно аномальні події, які не підпадають під звичний профіль мережевої активності. Наприклад, якщо в системі SIEM реєструється незвично висока активність в неробочий час, метод може ізолювати ці події як потенційні аномалії, що потребують додаткового аналізу.

Наступний метод заснований на стандартній логістичній регресії, яка є одним із популярних алгоритмів для задач класифікації. Логістична регресія оцінює ймовірність належності об’єкта (наприклад, мережевої події) до певного класу, використовуючи логістичну функцію. Проте, у підході вводиться адаптивна зміна ваг кожної ознаки, що дозволяє враховувати поточний стан мережі, особливості трафіку й інші фактори. Зазвичай логістична регресія використовує фіксовані ваги для кожної ознаки, проте в даному випадку ваги будуть динамічно змінюватися на основі частоти певних типів подій або аномальних ознак. Це забезпечить гнучке реагування на нові загрози.

Спочатку модель логістичної регресії налаштовується на навчальному наборі даних з використанням класичних методів, таких як градієнтний спуск, щоб визначити початкові ваги ознак. Система постійно моніторить мережеві події. На основі аналізу останніх подій відстежуються певні патерни, наприклад, частота з’явлення подій з певного IP-діапазону або з певного порту. Кожній ознаці (IP-адреса, час, тип події) присвоюється вага, яка залежить від її поточної важливості в умовах останніх мережевих подій.

Якщо виявляється, що подія з певною ознакою (наприклад, IP-адреса або тип запиту) з’являється частіше, ніж очікувалося, система підвищує вагу цієї ознаки, щоб зробити її більш значущою для наступних класифікацій. Навпаки, для менш частих або рідкісних ознак вага може зменшуватись. Це дозволяє адаптуватися до аномалій у реальному часі, збільшуючи ймовірність правильного виявлення загроз.

В міру надходження нових даних модель може автоматично коригувати ваги ознак без повного перенавчання. Це реалізується за допомогою адаптивного коригування ваг у реальному часі, щоб модель завжди була «актуальною» й відповідала поточним загрозам. Наприклад, у разі різкого зростання подій із

певного IP-адреси модель збільшить вагу цієї ознаки для того, щоб розпізнавати такі події як потенційні загрози. Використовуючи модифіковані ваги ознак, модель визначає ймовірність того, що подія є аномальною або нормальною, і приймає рішення щодо класифікації на основі адаптивних ваг.

Цей метод дозволяє динамічно адаптуватися до нових патернів мережевої активності й враховувати нестабільність у поведінці мережі. Адаптація ваг дозволяє використовувати логістичну регресію для аналізу подій у режимі реального часу, забезпечуючи більш високу точність класифікації нових типів аномалій та загроз.

Алгоритм роботи цього методу буде виглядати наступним чином:

1) алгоритм починається з ініціалізації основних параметрів і первинного навчання логістичної регресії на історичних даних;
2) надходять нові дані про мережеві події, які містять такі ознаки, як IP-адреси, час запиту, тип події тощо;
3) дані очищаються та перетворюються в числовий формат для використання в моделі. На цьому етапі можна також стандартизувати дані;
4) система оцінює частоту повторення окремих ознак, щоб виявити зміни в характері мережевого трафіку. Так оцінюється кількість запитів з конкретних IP-адрес або за певними параметрами трафіку, типи та кількість запитів до серверів. Аналіз частоти допомагає виявити, які ознаки мають відхилення від норми, наприклад, коли певна IP-адреса надто часто генерує запити або коли незвично велика кількість з'єднань надходить за короткий час;
5) ознаки, які зустрічаються частіше або менш передбачувано, ідентифікуються як потенційно важливі для наступних класифікацій. Визначаються значущі відхилення, коли частота або інші характеристики ознаки виходять за рамки встановлених порогів за допомогою персентилів;

6) для ознак з аномальною частотою або частотою, що перевищує певний поріг, автоматично підвищується вага, щоб посилити їхній вплив на рішення моделі. Так, якщо ознака має аномально високу частоту або інші характерні відхилення, їй надається підвищена вага. Ваги ознак можуть змінюватися в залежності від частоти і значущості, що дозволяє алгоритму швидше реагувати на поточні аномалії. А в варіанті коли частота виникнення певної ознаки знижується до стандартних значень, вага може бути зменшена, щоб не впливати на загальний результат класифікації. Коригування ваг можна виконувати з певною періодичністю або в режимі реального часу, що дозволяє адаптуватися до поточних змін в мережевій активності;
7) з використанням адаптованих ваг модель логістичної регресії визначає ймовірність належності події до класу аномалій або нормальної активності. Рішення приймається на основі порогу класифікації, що залежить від ваг;
8) якщо подія класифікована як аномальна, її додають до списку для подальшого моніторингу або вживають відповідних заходів (наприклад, надсилають сигнал тривоги в SIEM-систему).

Рисунок 2.4 – Алгоритм роботи модифікованої логістичної регресії з адаптивними вагами

Останнім методом керованого навчання є K-найближчі сусіди (KNN) з індивідуальними вагами для кожного сусіда. Це адаптація класичного алгоритму, що дозволяє покращити точність класифікації за рахунок різного впливу кожного сусіда на прийняття рішення.

Підхід корисний для виявлення аномалій або класифікації мережевих подій, коли деякі сусіди мають більшу вагу через близькість або інші характеристики. У класичному KNN кожен із K найближчих сусідів має рівну вагу, і рішення приймається на основі більшості класів серед сусідів. Однак, у варіанті з індивідуальними вагами кожному сусіду призначається вага, що залежить від відстані до тестової точки або інших характеристик. Найближчі сусіди отримують вищу вагу, а віддаленіші — меншу, що дозволяє алгоритму точніше оцінювати ймовірність приналежності тестової точки до певного класу.

Для кожного тестового зразка обчислюються відстані до всіх точок у навчальному наборі, і вибираються K найближчих точок. Відстань зазвичай вимірюється за допомогою метрики, використовується Евклідова. Кожному з K сусідів присвоюється вага, яка визначає його вплив на рішення. Вага може залежати від:

- чим ближче сусід до тестової точки, тим більшу вагу йому надають. Часто використовуються зворотні значення відстані (наприклад, вага $= \frac{1}{d}$);
- якщо відомо, що певні точки є більш значущими, їм можна надати вищу вагу. Наприклад, сусіди, які вже позначені як аномальні, можуть отримувати нижчі ваги.

Ваги сусідів групуються за класами. Для кожного класу обчислюється сума ваг, після чого клас із найбільшою сумою ваг призначається тестовій точці. Тестовій точці присвоюється клас, який має найбільшу загальну вагу серед сусідів. Це враховує як кількість сусідів кожного класу, так і відстані до них.

Для цього алгоритму блок схема буде виглядати наступним чином:

1) отримання вхідних даних, де є тестова точка і навчальна вибірка з позначеними подіями;
2) обчислення відстаней між тестовою точкою і всіма точками навчальної вибірки;
3) вибір K найближчих сусідів за найменшою відстанню;

4) призначення ваг кожному сусіду відповідно до відстані або інших характеристик;
5) групування сусідів за класами і підсумовування ваг для кожного класу;
6) присвоєння класу тестовій точці на основі суми ваг;
7) вихідний клас тестової точки (наприклад, аномальна або нормальна подія).

Рисунок 2.5 – Алгоритм роботи методу K-найближчі з індивідуальними вагами для кожного сусіда

2.2 Інтеграція з SIEM системою

Процес інтеграції передбачає побудову інтерфейсу між SIEM та засобом, у нього є наступні функції:

—для аналізу подій засіб отримує лог-файли та метадані безпосередньо з SIEM-системи через API;

—отримані дані обробляються із застосуванням машинного навчання, з метою ідентифікації аномальних подій, оцінки ризиків та виявлення можливих загроз;

—результати аналізу передаються назад до SIEM для подальшого використання. Це включає маркування подій, пріорітизацію ризиків, відправку сповіщень та виведення аналітичних графіків.

В залежності від можливостей SIEM-системи вибирається метод інтеграції. Більшість сучасних систем мають RESTful API, що дозволяє легко підключитися до них для обміну даними, тому використовуємо саме його.

Для забезпечення безпечного обміну даними між SIEM системою та засобом передаються ключі, що запобігає

Залежно від вимог до швидкості обробки та обсягу даних, необхідно налаштувати параметри запитів API, наприклад, обсяг запитів, обмеження часу, чергу обробки тощо.

Після всіх налаштувань та встановлення з’єднання SIEM система ініціалізує процес збору даних з мережі системи. Основні етапи цього алгоритму включають такі кроки:

1) спочатку SIEM система визначає джерела подій, що охоплюють мережеві пристрої а саме маршрутизатори, комутатори, сервери, бази даних, програми, брандмауери. Для кожного з цих пристроїв визначаються логи, що мають бути зібрані для аналізу;
2) далі SIEM-система налаштовується на регулярний збір даних із кожного з підключених джерел. Це відбувається через використання агентів, що встановлені на окремих пристроях;

3) зібрані логи надходять у центральну базу даних SIEM-системи. Цей етап відбувається у режимі реального часу для важливих подій та через регулярні інтервали для менш критичних подій. Логи включають інформацію про підключення, час входу в систему, операції з файлами, зміни конфігурацій та інші критичні дії;
4) оскільки різні пристрої генерують логи у різних форматах, SIEM-система застосовує нормалізацію, щоб привести дані до єдиного стандарту. Це спрощує їх подальший аналіз та співставлення. Нормалізація включає виділення основних полів, таких як IP-адреси, часові мітки, коди помилок, і стандартизацію їх структури. Також вона необхідна для використання для подальшого аналізу засобом, так як дані мають підлягати одному вигляду;
5) після нормалізації система агрегує події, що дозволяє зменшити обсяг даних, виділивши лише унікальні або значущі події. Це допомагає уникнути дублювання та знижує навантаження на систему, зберігаючи тільки важливу інформацію;
6) на цьому етапі система застосовує фільтри для відсіювання подій, які не мають значення для аналізу безпеки. Наприклад, стандартні операції можуть бути відсортовані, а події, що пов'язані з підозрілою активністю, маркуються для детальнішого аналізу. Категоризація дозволяє групувати події за типом, такі як події входу/виходу, мережеві з'єднання, або доступ до файлів;
7) після фільтрації важливі дані зберігаються у центральній базі даних для подальшого аналізу. Архівування допомагає зберігати історичні дані, які можуть бути використані для ретроспективного аналізу або у проведенні розслідувань.

Рисунок 2.6 – Алгоритм збору даних SIEM-системою

Коли SIEM-система отримує дані від алгоритмів машинного навчання, ці дані можуть включати передбачення, оцінки аномальності або інші аналітичні показники, що допомагають виявляти загрози на основі історичних або нових патернів поведінки. Ось як виглядає алгоритм обробки цих даних SIEM-системою:

1) алгоритми машинного навчання аналізують мережеві події та видають оцінки аномальності такі як відсоток ймовірності загрози і класифікують події за категоріями ризику. SIEM-система приймає ці результати у вигляді числових показників, етикеток або сигналів з високим рівнем ймовірності для потенційних загроз;
2) SIEM-система адаптує дані від машинного навчання до свого внутрішнього формату, щоб забезпечити інтеграцію з іншими подіями та сигналами. Це включає стандартизацію оцінок аномальності та переведення їх у шкалу ризику, прийнятну для SIEM;
3) за допомогою попередньо визначених порогів та правил система відсіває малозначні події. Так якщо модель визначає аномалію як низькоризикову, SIEM може ігнорувати її або позначити як подію, що не потребує негайного втручання;
4) результати від машинного навчання об’єднуються з іншими подіями, що надходять із мережевих пристроїв, систем безпеки або журналів, зібраних SIEM. Це дозволяє побачити загальну картину та виявляти більш комплексні загрози. До прикладу, низка низькоризикових подій, які самі по собі не є загрозою, може вказувати на складну атаку, коли аналізується у поєднанні з іншими даними;
5) SIEM виконує кореляцію, використовуючи отримані оцінки аномальності для встановлення зв'язків між подіями з різних джерел. Це дозволяє визначити шаблони, які можуть вказувати на складні атаки. Наприклад, SIEM може поєднувати результати від моделей із певними ланцюжками подій (підозрілими IP-адресами або незвичним трафіком) і сигналізувати про загрозу;

7) на основі оцінок машинного навчання та кореляції SIEM-система визначає пріоритет для кожного інциденту. Інциденти високого пріоритету можуть отримувати статус критичних і передаватися на опрацювання аналітикам безпеки або автоматично запускати сценарії реагування;
8) SIEM може періодично аналізувати аномалії, які підтвердилися як реальні загрози, і передавати ці дані для подальшого тренування ML-моделей. Це дає змогу адаптувати моделі під нові загрози та підвищувати їх точність. SIEM-система оновлює моделі, використовуючи нові дані, зокрема нові шаблони атак або кореляції, що раніше не були враховані;
9) у випадках виявлення високоризикових загроз SIEM система може автоматично ініціювати захисні заходи, такі як блокування підозрілого IP-адреса, відключення облікових записів або обмеження доступу. Це дозволяє швидко реагувати на загрози та знижувати потенційні ризики для мережі;
10) всі події та їх обробка системою документуються для створення звітів, які можуть бути проаналізовані фахівцями з безпеки для подальшого вдосконалення процесів. У звітах також можна оцінити ефективність моделей машинного навчання, які допомагають виявляти загрози.

Рисунок 2.7 – Алгоритм дій SIEM-системи після отримання даних про події

Цей алгоритм інтеграції даних машинного навчання із SIEM дозволяє покращити точність виявлення загроз та знижує кількість хибних спрацювань. Такий підхід забезпечує адаптивність системи до нових типів загроз і змін у мережевих патернах.

3 ЗАСІБ АНАЛІЗУ МЕРЕЖЕВИХ ПОДІЙ ДЛЯ SIEM СИСТЕМИ

3.1 Засоби розробки

Для виконання поставлених задач в розробці засобу було обрано мову програмування Python. Цей вибір зумовлений низкою переваг, які роблять Python ідеальним інструментом для розробки в області аналізу даних, машинного навчання та систем інформаційної безпеки.

Він має простий і зрозумілий синтаксис, що прискорює розробку та зменшує кількість помилок у коді. Завдяки цьому, мова дозволяє зосередитися на логіці програми, а не на складності синтаксису. Ця мова програмування широко використовується у галузях аналізу даних, кібербезпеки та автоматизації. Це забезпечує доступ до великої кількості документації, прикладів коду та підтримки з боку спільноти. Python надає можливість легко інтегруватися з базами даних, системами управління інцидентами (наприклад, SIEM-системами) і зовнішніми API, що є важливим у контексті цієї роботи.

Цією мовою надається до використання широкий набір бібліотек для роботи з даними та реалізації алгоритмів машинного навчання. Це дозволяє мінімізувати час і зусилля, необхідні для реалізації складних функціональних можливостей. Серед бібліотек які використовуються для засобу можна виокремити наступні:

1) NumPy - ця бібліотека використовується для виконання базових операцій над даними, таких як нормалізація, обчислення відстаней між точками, та підготовка даних для алгоритмів класифікації.. Її вибір зумовлений високою швидкістю роботи завдяки використанню оптимізованого C-коду і зручною роботою з масивами та матрицями даних, що є основою для реалізації алгоритмів машинного навчання. Також NumPy надає широкий набір математичних функцій для операцій із векторами та матрицями.

2) Pandas — бібліотека для маніпуляції даними у табличному форматі, яка дозволяє ефективно працювати з великими обсягами інформації. Pandas використовується для обробки даних, отриманих із систем моніторингу, включаючи підготовку даних до аналізу, відбір релевантних характеристик і генерацію наборів для навчання моделей. Ця бібліотека має інтуїтивний інтерфейс для обробки даних у вигляді DataFrame та підтримує роботу з великими наборами даних, у тому числі фільтрацію, агрегацію та перетворення. Також Pandas надає можливість читання й запису даних у різних форматах, таких як CSV, Excel, SQL тощо.
3) Scikit-learn це одна з найпопулярніших бібліотек для машинного навчання. Вона забезпечує готові до використання алгоритми класифікації, регресії, кластеризації та роботи з даними. Ця бібліотека використовувалася для побудови моделей класифікації та виявлення аномалій. Наприклад, для роботи з алгоритмом Isolation Forest для ідентифікації аномальних мережевих подій. Основними плюсами цієї бібліотеки є простота використання та висока якість реалізації алгоритмів, підтримка широкого спектру методів, таких як логістична регресія, K-найближчі сусіди, Isolation Forest тощо, а також наявність вбудованих інструментів для передобробки даних (нормалізація, стандартизація).
4) Matplotlib є універсальною бібліотекою для створення графіків і візуалізації даних. Вона використовувалася для створення графіків класифікації, візуалізації розподілу даних, а також демонстрації роботи алгоритмів. Бібліотека має широкі можливості для налаштування графіків, що дозволяє створювати візуалізації високої якості. Також вона підтримує інтеграцію з іншими бібліотеками, такими як NumPy та Pandas і головне дозволяє відображати результати роботи алгоритмів у вигляді зрозумілих графічних представлень.

3.2 Програмна реалізація

Засіб починає свою роботу з налаштування з’єднання з SIEM системою для збору логів та інформації про мережу. Для цього встановлюється підключення по IP та логіну й паролю користувача агента.

```
WAZUH_API = "https://<WAZUH_MANAGER_IP>"
    USER = "<USER>"
    PASSWORD = "<PASSWORD>"
```

Далі відбувається запит через API та спроба встановити з’єднання, якщо воно успішне користувач отримує повідомлення про активне підключення

```
    response = requests.get(
        f"{WAZUH_API}/alerts?agent_id=<AGENT_ID>",
        auth=HTTPBasicAuth(USER, PASSWORD),
        verify=False
    )
      if response.status_code == 200:
        print("Connection successful!")
        print(response.json())
    else:
        print(f"Error: {response.status_code}")
```

Після підключення SIEM система проводить збір даних з пристрою через агента та відбувається їх нормалізація у функції file_processing(). Спочатку вибираються колонки з датасету які будуть використовуватись.

```
selected_columns = [
        "dur", "spkts", "dpkts", "sbytes", "dbytes", "rate",
        "sload", "dload", "sttl", "dttl", "sinpkt", "dinpkt",
        "tcprtt", "synack", "ackdat", "proto", "service", "state"
    ]
    data_selected = data[selected_columns]
```

Визначаються категорійні зміні які потрібно закодувати і перетворюються для подальшого використання

```
categorical_columns = ["proto", "service", "state"]
    label_encoders = {}
    for column in categorical_columns:
        le = LabelEncoder()
        data_selected[column]                                                    =
le.fit_transform(data_selected[column].astype(str))
        label_encoders[column] = le
```

Числові ознаки масштабуються та декілька рядків виводиться для перевірки правильності оформлення

```
numerical_columns = [col for col in selected_columns if col not in categorical_columns + ["label"]]
    scaler = StandardScaler()
    data_selected[numerical_columns] = scaler.fit_transform(data_selected[numerical_columns])
print("Перші кілька рядків підготовлених даних:")
    print(data_selected.head())
```

Якщо дані ще будуть використовуватись для тренування моделі то вони зберігаються і встановлюються мітки

```
X = data_selected.drop(columns=["label"])  # Дані для тренування
    y = data_selected["label"]  # Мітки
output_path = "preprocessed_data.csv"
    data_selected.to_csv(output_path, index=False)
    print(f"Підготовлені дані збережено у файл: {output_path}")
```

Якщо будь якій з функцій необхідно надати дані для роботи або тренування то вони завантажуються з файлу через окрему функцію, при цьому дані мають бути вже попередньо оброблені та приведені до нормалізованого вигляду.

```
def file_loader():
    file_path = "preprocessed_data.csv"
    data = pd.read_csv(file_path)
    return data
```

Тепер проведемо опис як працює функція кластеризації k-means. Спочатку визначаються кількість кластерів необхідна для роботи, створюється модель та зберігаються координати центроїдів

```
kmeans = KMeans(n_clusters=10)
kmeans.fit(network_data)
labels = kmeans.labels_
centroids = kmeans.cluster_centers_
```

Потім для кожної точки обчислюється евклідова відстань до центроїду відповідного кластеру. Це дозволяє оцінити, наскільки точка "далека" від центру свого кластеру.

```
distances = np.linalg.norm(network_data - centroids[labels], axis=1)
```

Далі визначається порогове значення яке обчислюється відносно відстані точок які належать до кластеру. Всі 90% точок що знаходяться ближче до центроїду визначаються як нормальні а 10% як потенційно аномальні. Значення порогу для кожного кластеру зберігається у словнику thresholds.

```
thresholds = {}
for i in range(len(centroids)):
    cluster_distances = distances[labels == i]
    threshold = np.percentile(cluster_distances, 90)  # 90-й персентиль
    thresholds[i] = threshold
```

Для кожної точки перевіряється, чи її відстань до центроїду перевищує порогове значення для її кластеру. Якщо відстань більше за поріг, точка вважається аномальною. Індекси таких точок додаються до списку anomalies.

```
anomalies = []
for i, distance in enumerate(distances):
    cluster_id = labels[i]
    if distance > thresholds[cluster_id]:
        anomalies.append(i)
anomalies = np.array(anomalies)
```

Вибираються координати аномальних точок із початкового набору даних.

```
anomalies_2d = network_data.iloc[anomalies]
```

Після цього відбувається візуалізація кластерів з аномальними, нормальними точками та центроїдами. Сині точки – це нормальні точки (усі дані, крім аномальних). Червоні точки – аномальні дані, які виходять за межі порогового значення. Зелені "хрестики" – центроїди кластерів.

```
plt.scatter(network_data.iloc[:, 0], network_data.iloc[:, 1], c='blue', label='Нормальні точки')
plt.scatter(anomalies_2d.iloc[:, 0], anomalies_2d.iloc[:, 1], c='red', label='Аномальні точки')
plt.scatter(centroids[:, 0], centroids[:, 1], c='green', marker='x', label='Центроїди')
```

Для кожної точки проводиться сіра пунктирна лінія до її центроїду. Це дозволяє візуально оцінити, наскільки далеко точка знаходиться від центру кластеру.

```
for i in range(len(network_data)):
    plt.plot([network_data.iloc[i, 0], centroids[labels[i], 0]],
             [network_data.iloc[i, 1], centroids[labels[i], 1]],
             c='gray', linestyle='dashed', alpha=0.5)
```

Додаються підписи до осей, легенда, сітка та заголовок для покращення розуміння графіка.

```
plt.xlabel('Ознака 1')
plt.ylabel('Ознака 2')
plt.legend()
plt.grid(True)
plt.title('Візуалізація кластерів з аномаліями та лініями до центроїдів')
plt.show()
```

Ця функція допомагає сегментувати дані на кластери за допомогою алгоритму K-Means, виявити аномалії на основі відстаней до центроїдів та візуалізувати кластери, аномалії та відстані між точками й центроїдами, щоб проаналізувати, які точки відхиляються від загального розподілу.

Наступна функція isolation_forest_labels виконує аналіз даних для виявлення аномалій за допомогою алгоритму Isolation Forest і візуалізує результати. На початку роботи вона визначає матрицю ознак X та мітки аномалій Y.

```
X = network_data_scaled.drop(columns=[label_column])
y = network_data_scaled[label_column]  # Мітки аномалій (0 - нормальні, 1 - аномальні)
```

Потім дані розбиваються на тренувальну (75%) та тестову (25%) вибірки. Параметр stratify=y забезпечує, щоб розподіл аномалій і нормальних точок був однаковим у тренувальних і тестових наборах.

```
X_train, X_test, y_train, y_test = train_test_split(X, y, stratify=y, random_state=42)
```

Наступний етап це зниження вимірності, а саме зменшення кількості ознак до двох головних компонент. Це полегшує візуалізацію багатовимірних даних. Тренувальні дані знижуються до двох компонент (X_train_2d), а тестові трансформуються так само (X_test_2d).

```
pca = PCA(n_components=2)
X_train_2d = pca.fit_transform(X_train)
X_test_2d = pca.transform(X_test)
```

Відображаються точки тренувальної вибірки, колір відповідає міткам: нормальні точки (0) та аномальні (1). Це дозволяє оцінити розподіл даних у просторі двох головних компонент.

```
plt.scatter(X_train_2d[:, 0], X_train_2d[:, 1], c=y_train, s=20, edgecolor="k")
```

Далі будується Isolation Forest для виявлення аномалій. Модель навчається на тренувальних даних (X_train).

```
clf = IsolationForest(max_samples=100, random_state=0)
clf.fit(X_train)
```

Алгоритм прогнозує мітки для тестових даних (y_pred), де -1 – аномальна точка, а 1 – нормальна точка. Масив anomalies визначає, які точки тестового набору є аномальними.

```
y_pred = clf.predict(X_test)
anomalies = y_pred == -1
```

Відображаються точки тестової вибірки, де колір вказує, чи є точка аномальною або нормальною, відповідно до прогнозів моделі. Це дозволяє оцінити, наскільки ефективно модель класифікує точки.

```
plt.scatter(X_test_2d[:, 0], X_test_2d[:, 1], c=anomalies, s=20, edgecolor="k")
plt.title("Результати Isolation Forest після зниження вимірності")
plt.xlabel("Перша головна компонента")
plt.ylabel("Друга головна компонента")
plt.legend(["Нормальні", "Аномальні"])
plt.show()
```

В результаті користувач отримує два графіки, один з початковим розподілом даних і другий з результатом роботи алгоритму з класифікацією точок. Функція дозволяє проаналізувати мережеві події, виділяючи потенційно аномальні точки. Завдяки візуалізації результатів, можна оцінити якість роботи моделі й визначити проблемні області в даних.

Клас AdaptiveLogisticRegression реалізує алгоритм логістичної регресії з адаптивними вагами. На початку роботи він ініціалізує дані необхідні для моделі, а саме швидкість навчання, визначає розмір кроку оптимізації, кількість ітерацій для оновлення ваг, регуляризація для запобігання перенавчанню (L2-регуляризація).

```
def __init__(self, learning_rate=0.01, iterations=1000, lambda_reg=0.01)
```

Метод fit відповідає за навчання моделі, де спочатку дані масштабуються для забезпечення стабільної роботи градієнтного спуску, щоб всі ознаки мали однаковий вплив.

```
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
```

Ініціалізуються ваги моделі, які оновлюються під час навчання, та адаптивні ваги, що модифікують швидкість оновлення ваг залежно від градієнта. Та дані розбиваються на тренувальні і тестові.

```
self.weights = np.zeros(X_scaled.shape[1])
self.adaptive_weights = np.ones(X_scaled.shape[1])
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.3, random_state=42)
```

І відбувається навчання моделі з отриманням різниці між прогнозами та реальними значеннями.

```
for i in range(self.iterations):
    predictions = self.predict_proba(X_train)
    error = predictions - y_train
```

Градієнтний спуск визначає напрямок зміни ваг для зменшення помилки. Ваги оновлюються з урахуванням адаптивного кроку, чим більший градієнт, тим більший приріст адаптивної ваги, що зменшує швидкість оновлення для стабілізації навчання. А регуляризація зменшує значення ваг, запобігаючи їх надмірному росту.

```
gradient = X_train.T.dot(error) / len(y_train) + self.lambda_reg * self.weights
self.weights -= self.learning_rate * gradient / (self.adaptive_weights + 1e-6)
self.adaptive_weights += gradient ** 2
```

На кожній ітерації обчислюється точність на тренувальних і тестових наборах. Точності зберігаються для візуалізації прогресу навчання.

```
train_pred = self.predict(X_train)
test_pred = self.predict(X_test)
self.train_accuracy.append(accuracy_score(y_train, train_pred))
self.test_accuracy.append(accuracy_score(y_test, test_pred))
```

Для прогнозування ймовірностей використовується сигмоїдна функція, яка обчислює ймовірності для кожного класу.

```
return 1 / (1 + np.exp(-X.dot(self.weights)))
```

Ймовірності перетворюються в класи, якщо ймовірність ≥ 0.5, точка належить класу 1 (аномальна). Інакше точка належить класу 0 (нормальна).

```
return (self.predict_proba(X) >= 0.5).astype(int)
```

Останнім етапом є обчислення втрат, де логістична функція втрат оцінює, наскільки добре модель відповідає реальним міткам. А регуляризація додається до втрат, щоб штрафувати великі значення ваг.

```
loss = -np.mean(y * np.log(predictions + 1e-6) + (1 - y) * np.log(1 - predictions + 1e-6))
regularization = self.lambda_reg * np.sum(self.weights ** 2) / 2
return loss + regularization
```

Використання індивідуальних ваг для кожної ознаки покращує стабільність і точність навчання. Алгоритм автоматично зменшує крок навчання для "складних" ознак (з великим градієнтом).

Клас WeightedKNN реалізує модифікацію методу k-найближчих сусідів де кожен сусід впливає на результат пропорційно своїй відстані до точки. На початку роботи відбувається ініціалізація параметрів, а саме k, яка відповідає за кількість найближчих сусідів, які використовуються для класифікації. Значення **k** визначає баланс між узагальненням і деталізацією, чим менше воно тим більше модель буде чутливою до шуму.

```
def __init__(self, k=5):
    self.k = k
```

Навчання у цьому методі просто зберігаються навчальні дані (X_train та y_train) для подальшого використання. Метод k-NN належить до інстанційних алгоритмів навчання, тому він не вимагає явного етапу тренування.

```
def fit(self, X_train, y_train):
    self.X_train = X_train
    self.y_train = y_train
```

Для кожної тестової точки виконується пошук найближчих сусідів у навчальному наборі.

```
def predict(self, X_test):
    predictions = []
    for i, x_test in enumerate(X_test):
```

Для обчислення відстаней використовується метрика евклідової відстані. Для тестової точки x_test обчислюються відстані до всіх точок навчального набору.

```
distances = pairwise_distances([x_test], self.X_train, metric='euclidean').flatten()
```

Після цього шукаються найближчі сусіди, де сортуються індекси відстаней за зростанням, вибираються індекси k найближчих точок і отримуються відповідні класи цих сусідів.

```
nearest_indices = np.argsort(distances)[:self.k]
nearest_classes = self.y_train[nearest_indices]
```

Ваги сусідів обчислюються як обернені значення їхніх відстаней, 1e-6 додається, щоб уникнути ділення на нуль, якщо тестова точка співпадає з навчальною.

```
weights = 1 / (distances[nearest_indices] + 1e-6)
```

Кожен сусід голосує за свій клас із вагою, пропорційною відстані, обчислюється зважена частота кожного класу. І argmax обирає клас із найбільшою зваженою частотою.

```
weighted_class = np.bincount(nearest_classes, weights=weights).argmax()
predictions.append(weighted_class)
```

accuracy_score обчислює відсоток правильних прогнозів.

```
accuracy = accuracy_score(y_test, y_pred)
```

Візуалізація відбувається через побудову графіка порівняння справжніх міток і передбачених значень.

```
plt.scatter(range(len(y_test)), y_test, color='blue', alpha=1, label="True labels")
plt.scatter(range(len(y_pred)), y_pred, color='red', alpha=0.5, label="Predicted labels")
```

3.3 Тестування засобу

Для тестування було розроблено графічний інтерфейс користувача у вигляді вікна на якому знаходяться декілька кнопок за допомогою яких можна вибирати потрібні функції програми або запускати їх по черзі для загального тестування (рис. 3.1).

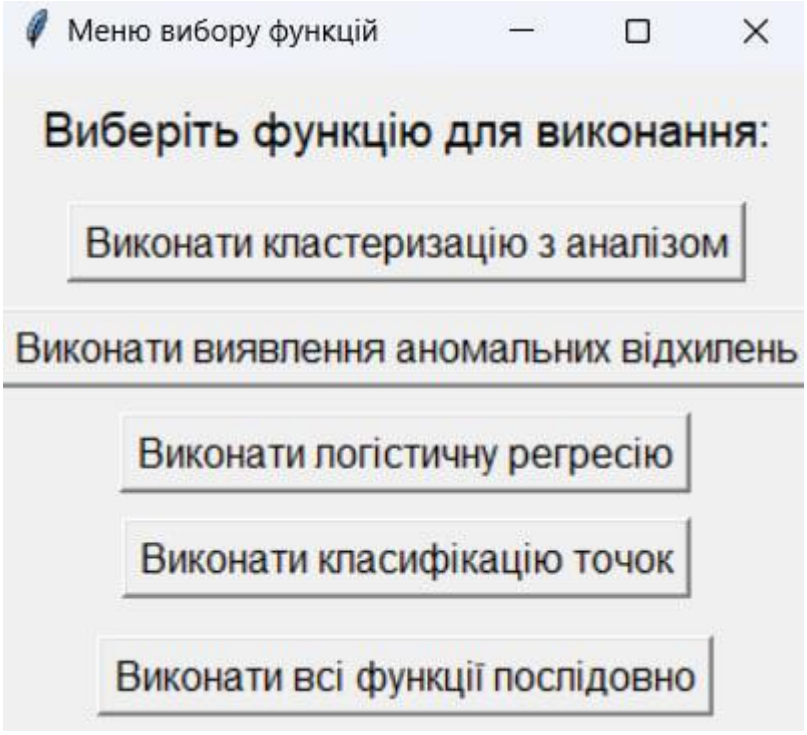


Рисунок 3.1 – Діалогове вікно взаємодії з програмою

Під час виконання будь якої функції з’явиться повідомлення або про успішне виконання, або про помилку, що буде свідчити про переривання роботи програми. Повідомлення про успішне виконання повідомляє яка саме функція закінчила свою роботу, це корисно при послідовному виконанні, так як користувач буде мати змогу зрозуміти на якому етапі роботи знаходиться засіб (рис. 3.2).

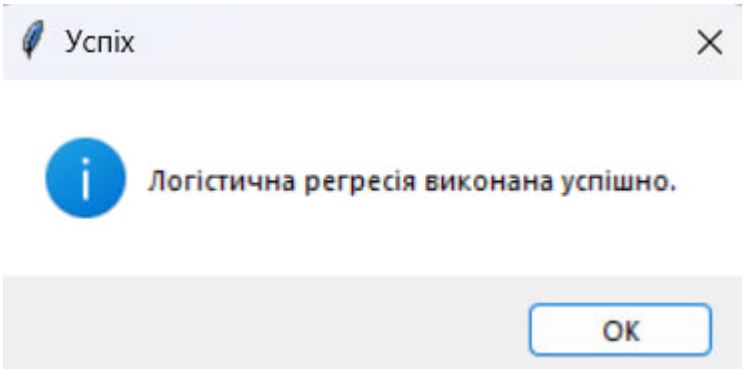


Рисунок 3.2 – Повідомлення програми про успішне виконання функції

Якщо ж під час роботи виникла якась помилка то програма повідомляє про це в відповідному вікні яке також наводить інформацію про саму помилку. Це допомагає користувачу краще зрозуміти суть проблеми та швидше знайти її можливе рішення (рис. 3.3).

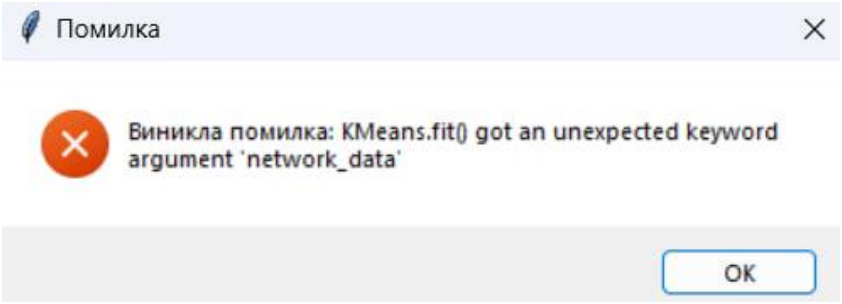


Рисунок 3.3 – Повідомлення програми про помилку під час виконання

На цьому графічний інтерфейс у вигляді кнопок для взаємодії закінчується, далі користувач отримує інформацію як у текстовому вигляді, так і у вигляді графіків або зображень у окремому вікні. На цих зображеннях відображаються дані, що були оброблені та позначені як нормальні або аномальні, також там може наводитись додаткова інформація в залежності від виконуваної функції. Вікна є інтерактивними, їх можна приближувати, переміщувати площину, зберігати зображення тощо.

Перша функція яку було протестовано це кластеризація, вона зображується у вигляді точок на площині, де за легендою сині точки це нормальні дані, червоні точки є аномальними а зеленим хрестом позначено центроїди кластерів.

Також на кластери впливають різні показники, такі як кількість цих кластерів та процентиль при якому дані вважаються аномальними. Так на рисунку 3.4 зображено 3 кластери при процентилі 90%.

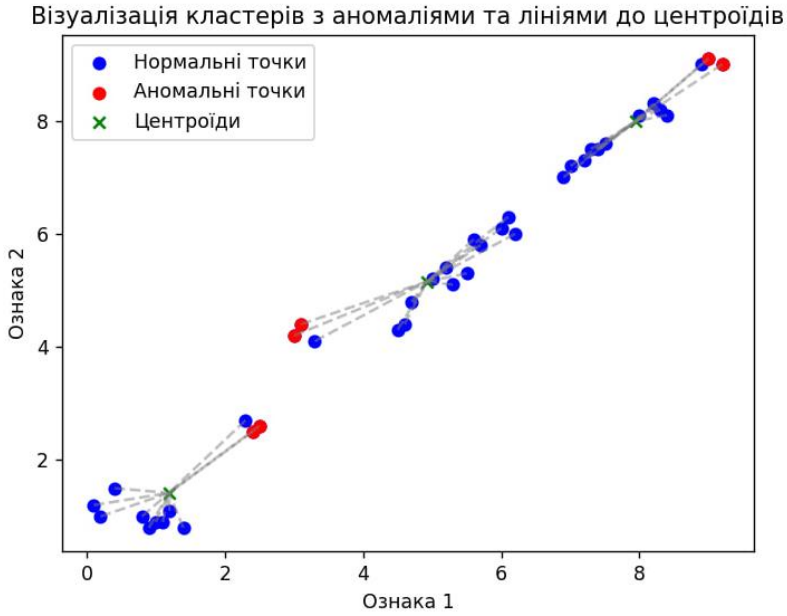


Рисунок 3.4 – Візуалізація 3 кластерів з 90% процентилем

Як видо на зображенні не всі точки які знаходяться віддалено від свого центроїду позначені як аномальні, тому точність не є 100%. При підвищенні процентилю до 95% кількість аномальних точок відповідно зменшується (рис. 3.5).

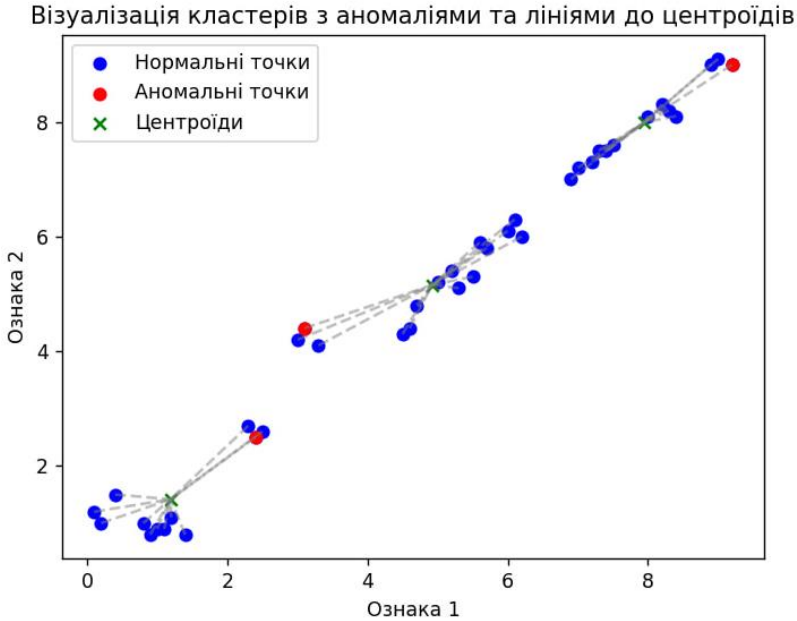


Рисунок 3.5 – Візуалізація 3 кластерів при 95% процентилі

Оптимальним варіантом для тестування є використання методу ліктя для розрахунку найкращої кількості кластерів для обсягу наявних даних. Так наприклад найбільш оптимальним варіантом після розрахунку для тестових даних є 4 кластери, для яких буде використовуватись 90% процентиль. Так дані не будуть скупчені в одну велику масу, через яку важко визначити дійсно не надто великі відхилення, а розбиті на декілька кластерів з схожими величинами (рис. 3.6).

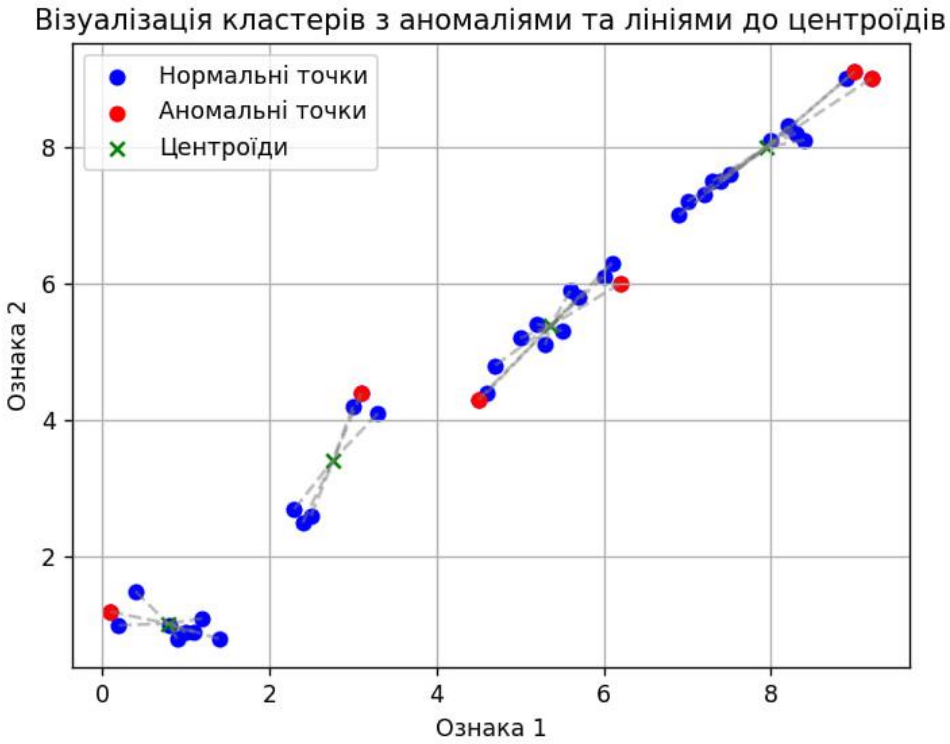


Рисунок 3.6 –Візуалізація 4 кластерів при 90% процентилі

Тестування на реальних даних показало що найкращим рішенням для аналізу є використання таких параметрів подій як кількість пакетів та розмір переданих даних в байтах. Так при тестуванні на більшій кількості подій було сформовано наступні зображення.

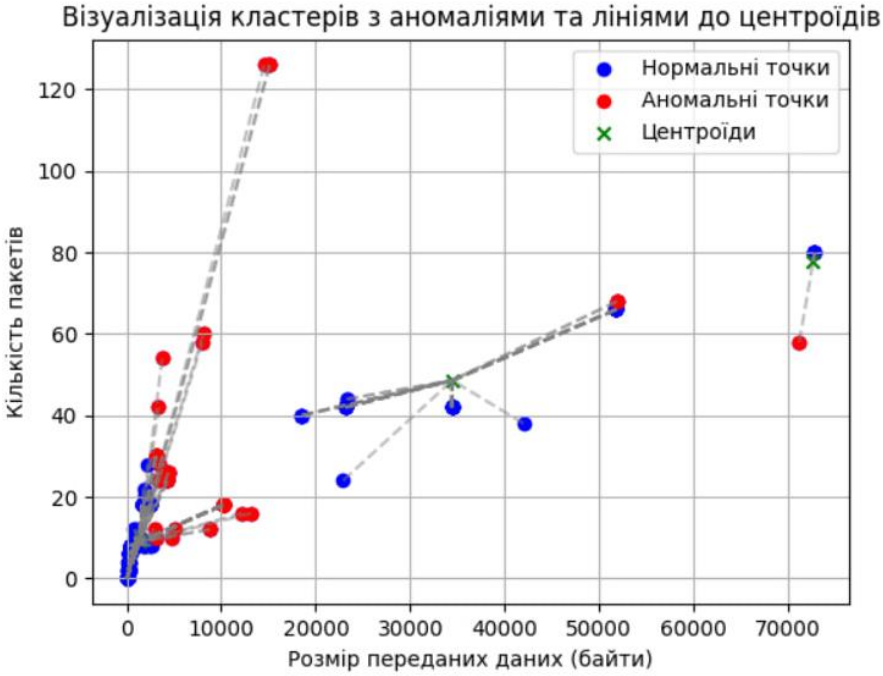


Рисунок 3.7 – Візуалізація 3 кластерів на 90% процентилі при великому обсягу даних

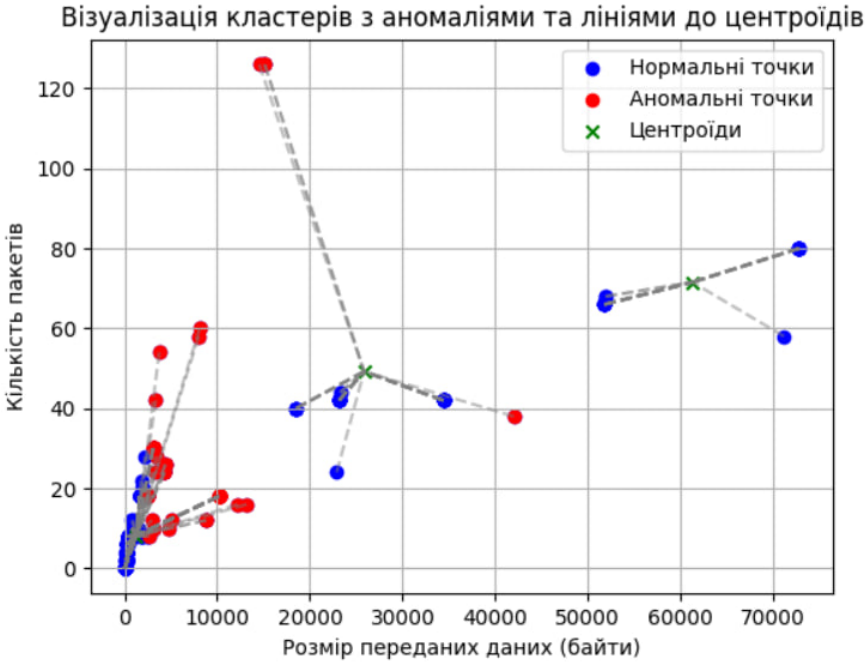


Рисунок 3.8 – Візуалізація 3 кластерів на 85% процентилі при великому обсягу даних

Наступним тестуванням буде тестування алгоритму виявлення аномалій на базі Isolation Forest. Алгоритм будує зображення точок на площині, де числові значення відповідають графіку. На зображенні жовтими точками позначено аномальні дані, які відрізняються від інших шляхом ізоляції дерев значень. Чорні точки це нормальні точки які не підпадають під визначення аномальності. Для

тестування роботи цього алгоритму насамперед потрібно визначити які параметри підпадають під його найкращі можливості, так було обрано часові параметри, зокрема тривалість з’єднання та інтервал між пакетами. На рисунках можна побачити як алгоритм ізолює точки в купу і виділяє ті що відрізняються від інших жовтим кольором, це вказує на те, що дані які відповідають цій координаті є аномальними (рис. 3.9).

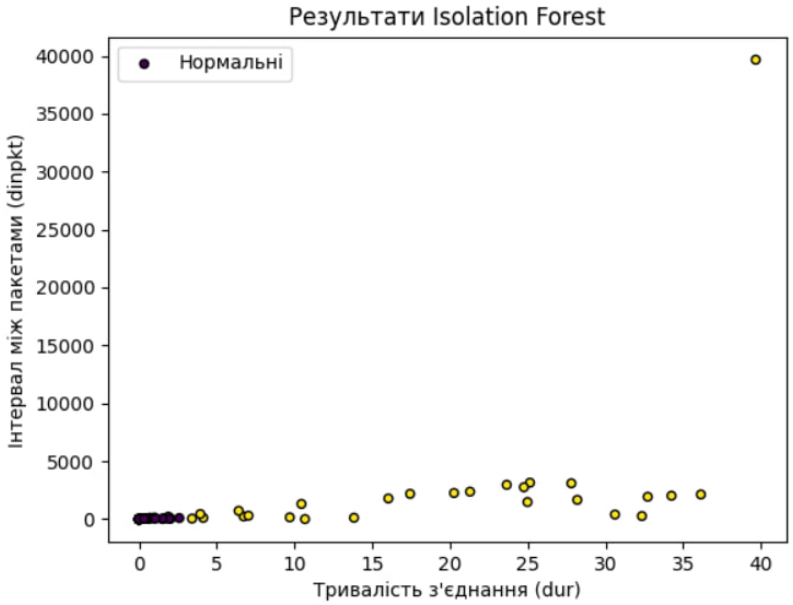


Рисунок 3.9 – Візуалізація виявлення аномальних даних шляхом ізоляції дерев

Тестування на інших наборах показує схожі результати, при цьому точність роботи алгоритму є доволі високою, він мало помиляється та вказує на аномальні події.

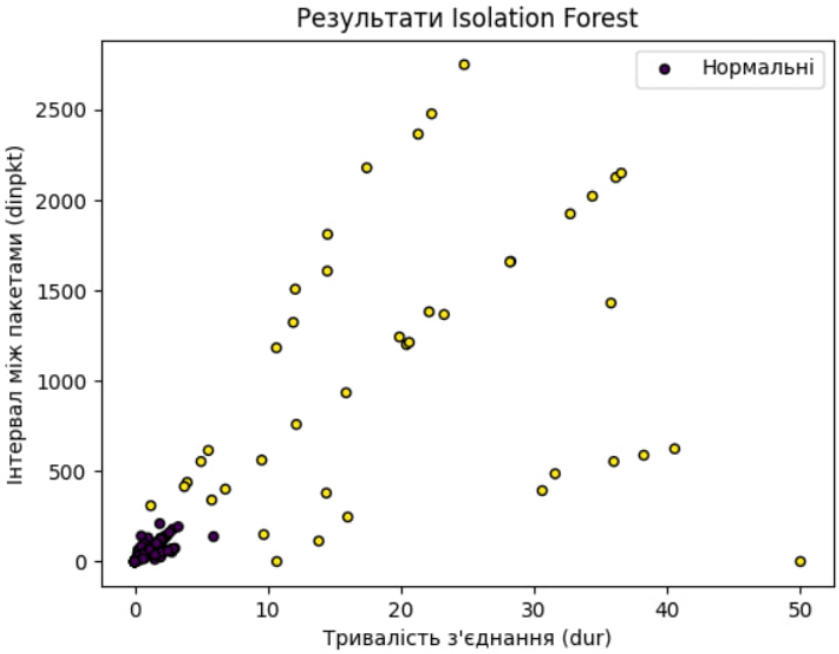


Рисунок 3.10 – Візуалізація виявлення аномальних даних шляхом ізоляції дерев

Наступною функцією програми буде модифікована логістична регресія. Вона не передбачає візуалізації окрім своєї точності моделі на тренувальних та тестових даних, де показує наскільки точно модель передбачає аномальність даних. Так на графіку представлено 2 лінії, перша є синьою і це точність на тренувальних даних, друга це помаранчева і вона вже показує точність тестових даних після тренування моделі. Так на площині є дві осі, вертикальна це точність від 0 до 1, де 0 – це 0%, а 1 – це 100%; горизонтальна вісь це ітерації які пройшла модель в кількісному значенні. Після тренування бачимо що точність цієї моделі складає приблизно 87,5% на тренувальних даних та 84,5% на тестових даних. Хоч точність і падає на тесті, але модель все ще показує непоганий результат роботи (рис. 3.9).

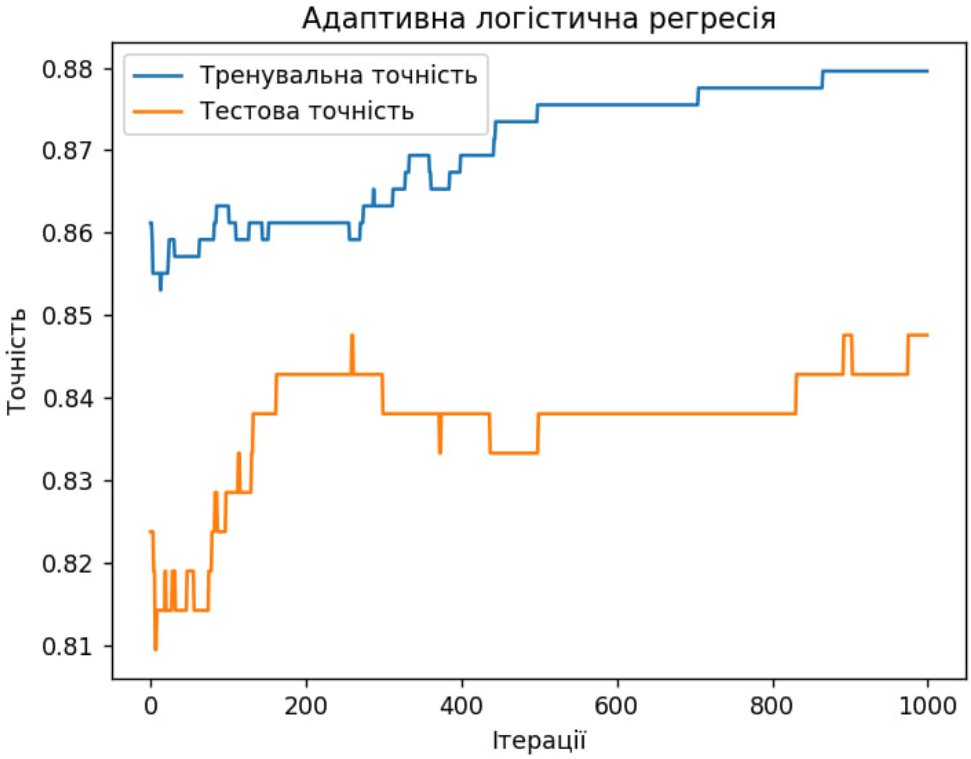


Рисунок 3.11 – Візуалізація точності моделі логістичної регресії з точністю 84,5%

Повторні тестування проводились на різних за обсягом даних, головною метою була перевірка чи не відбувається спадання точності при великому навантаженні, в результатах було отримано наступні графіки.

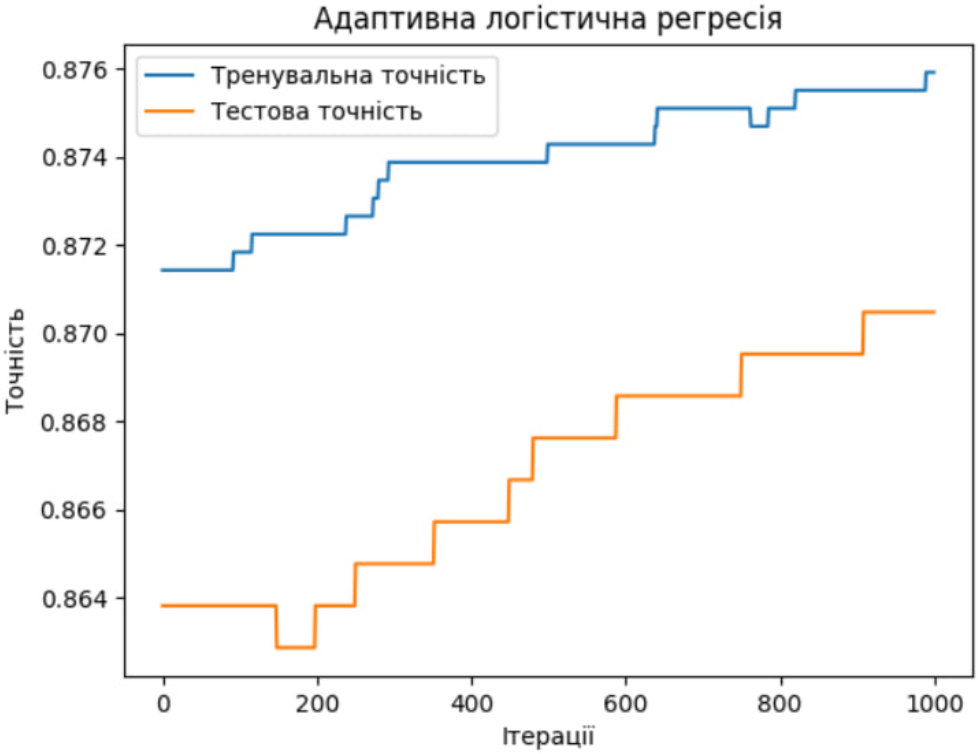


Рисунок 3.12 – Візуалізація точності моделі логістичної регресії з точністю 87%

Найкраще себе алгоритм показав на найбільшому обсягу даних (20000 записів), там він видав точність близько 88% (рис. 3.13).

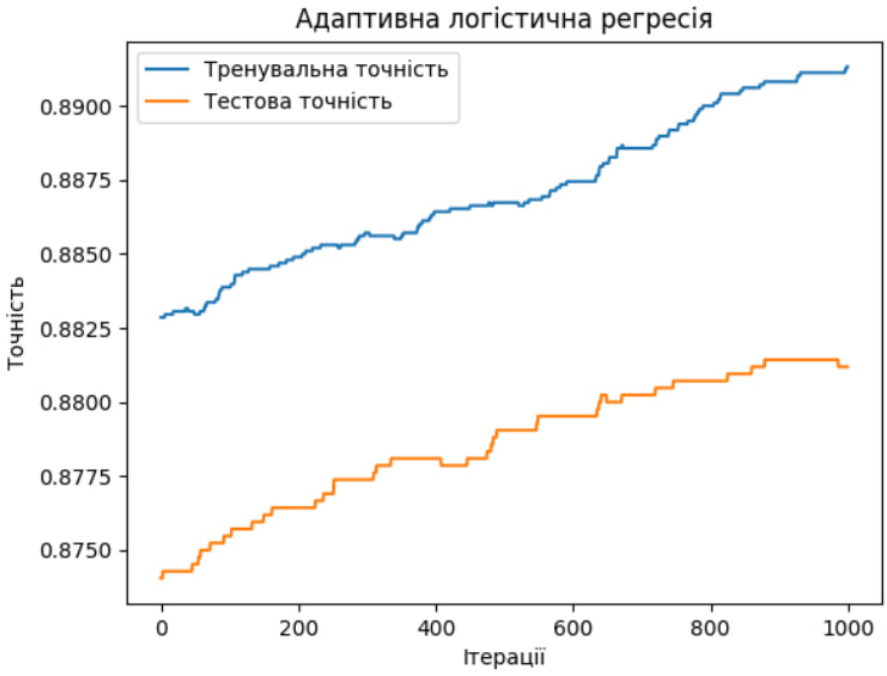


Рисунок 3.13 - Візуалізація точності моделі логістичної регресії з точністю 88%

Остання функція яка представлена в програмі це модель передбачення класифікацією даних. Вона реалізована в вигляді точок на площині, де є верхня межа та нижня межа. Це дані які відповідають значенням аномальності подій,

0 – нормальна подія, 1 – аномальна. Модель намагається передбачити якому типу відповідає той чи інший набір даних і накладає точки одна на одну. Якщо сині точки це справжні значення то червоні це передбачені. Так при накладанні червоної точки на синю рахується що модель передбачила її аномальність, на побудованій площині це зображено. Видно що точність моделі не є ідеальною, так як вона має 80,33% але швидкість роботи є доволі швидкою та обсяг даних для роботи доволі великим (рис. 3.10).

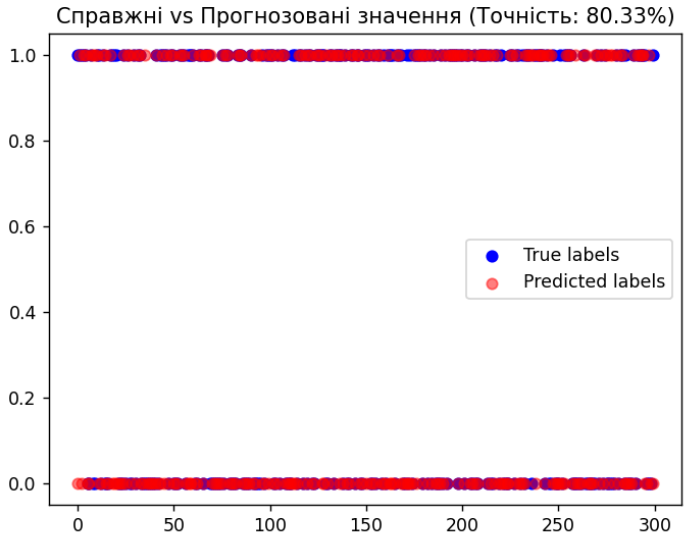


Рисунок 3.14 – Візуалізація прогнозу моделі шляхом класифікації даних з точністю 80,33%

При збільшенні обсягу тренувальних даних модель покращує свою точність, та прогнозує аномальні події з 95% точністю (рис. 3.15).

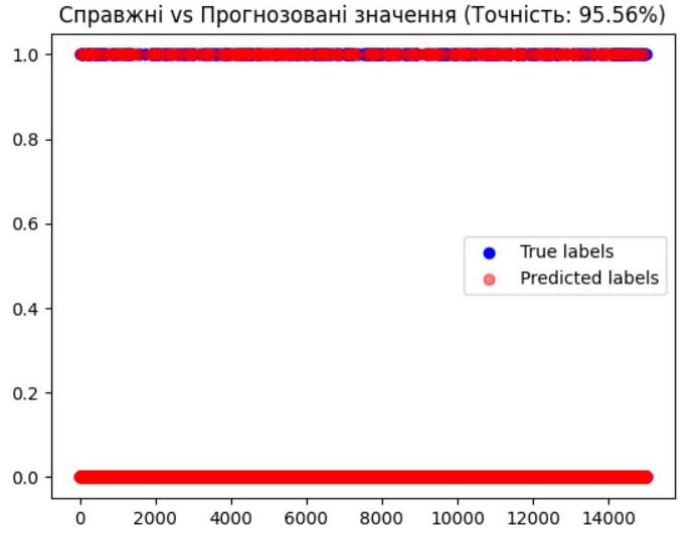


Рисунок 3.15 - Візуалізація прогнозу моделі шляхом класифікації даних з точністю 95,56%

Для кращого представлення візуальної частини цього алгоритму можна подивитись на гістограму, що побудована на основі цих даних, вона відображає на скільки модель робить правильні передбачення (рис. 3.16).

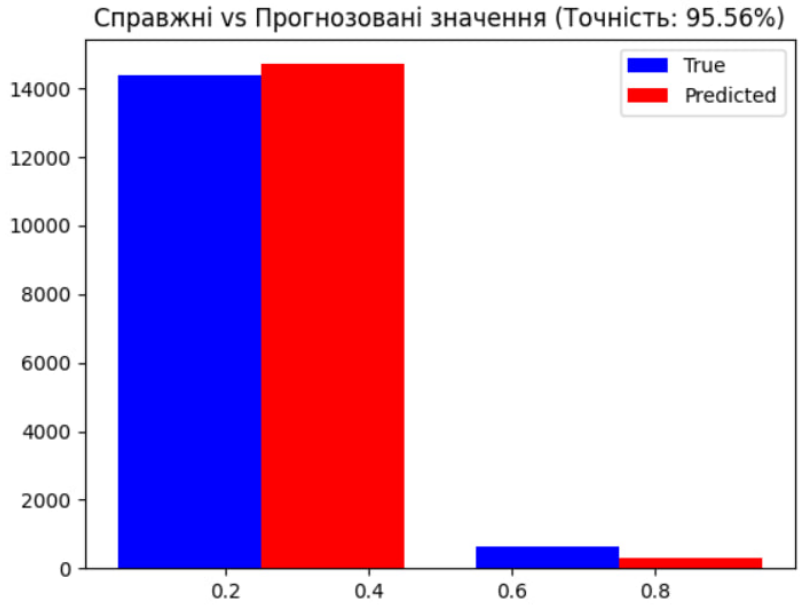


Рисунок 3.16 – Гістограма моделі класифікації даних

В результаті отримуємо засіб який не тільки описує дані в текстовому вигляді а й візуалізує їх у вигляді графіків, площин з точками. Це допомагає користувачу краще розуміти суть роботи засобу і покращує аналіз мережевих подій, що відповідає суті роботи.

Таблиця 3.1 – Порівняння існуючих аналогів з розробленим засобом

Параметр	**Розроблений засіб**	**Wazuh**	**Splunk**
Аналіз аномалій	Isolation Forest із динамічними порогами	Базовий сигнатурний аналіз	Автоматичний, але потребує ручного налаштування
Кластеризація	K-Means для групування та виявлення аномалій	Відсутня	Можливе через сторонні модулі

Параметр	Розроблений засіб	Wazuh	Splunk
Класифікація	Логістична регресія та KNN для уточнення результатів	Відсутня	Вбудована класифікація
Адаптивність	Динамічні ваги для врахування змін у трафіку	Фіксовані правила	Частково адаптивна
Точність роботи	**Висока (~90%)**, завдяки поєднанню алгоритмів класифікації та кластеризації	Середня (~70%), залежить від правил	Висока (~85%), але залежить від налаштувань
Швидкість обробки	**Висока**, оптимізована для великих обсягів даних	Середня, повільніша на великих наборах	Низька при великому обсязі даних
Вартість	Безкоштовне, базується на Python	Безкоштовне	Висока вартість ліцензії
Гнучкість у налаштуванні	Повна гнучкість для додавання нових алгоритмів	Обмежена	Обмежена

В результаті отримано засіб, що має більш розширений функціонал, базується на безкоштовному ПЗ, завдяки комбінації алгоритмів краще виявляє аномалії, особливо нові та нестандартні. Також підходить для великих обсягів даних завдяки оптимізації під конкретні задачі та використанню легких моделей.

4 ЕКОНОМІЧНА ЧАСТИНА

4.1 Оцінювання рівня конкурентоспроможності розробки

Конкурентоспроможність розкривається через систему якісних та економічних показників.

Для проведення оцінювання необхідно отримати оцінки експертів відповідно кожного критерія і обчислити середню арифметичну оцінку, яка визначатиме науково-технічний рівень та комерційний потенціал розробки.

Експертами виступають:

1) Лукічов Віталій Володимирович, доцент кафедри Захисту Інформації, ВНТУ;
2) Майданюк Володимир Павлович, доцент кафелри Програмного Забезпечення, ВНТУ;
3) Каплун Валентина Аполінаріївна, старший викладач кафедри ЗІ, ВНТУ.

Оцінювання проведено за показниками, що наведені в Додатку, оцінки цих показників, виражені в балах, представлені в таблицях 4.1 і 4.2.

Таблиця 4.1 - Результати оцінювання науково-технічного рівня і комерційного потенціалу розробки

Критерії	Експерт (ПІБ, посада)		
	Лукічов В.В.	Майданюк В.П.	Каплун В.А.
	Бали:		
1. Технічна здійсненність концепції	2	2	2
2. Ринкові переваги (наявність аналогів)	2	2	2
3. Ринкові переваги (ціна продукту)	2	2	2
4. Ринкові переваги (технічні властивості)	2	3	3
5. Ринкові переваги (експлуатаційні витрати)	3	2	2
6. Ринкові перспективи (розмір ринку)	2	2	2
7. Ринкові перспективи (конкуренція)	1	1	2
8. Практична здійсненність (наявність фахівців)	3	2	2
9. Практична здійсненність (наявність фінансів	2	2	2
10. Практична здійсненність (необхідність нових матеріалів)	4	3	4
11. Практична здійсненність (термін реалізації)	3	3	3
12. Практична здійсненність (розробка документів)	4	4	4
Сума балів	30	28	30
Середньоарифметична сума балів СБс	29,33		

$$СБ_С = \frac{\sum_{i=1}^{3} СБ_i}{3} = \frac{30+28+30}{3} = 29, \quad (4.1)$$

де:

$СБ_С$ – середньоарифметична сума балів

$СБ_i$ – сума балів і-го експерта

За результатами розрахунків, наведених в таблиці 4.1, робиться висновок щодо науково-технічного рівня і рівня комерційного потенціалу розробки. При цьому використовують рекомендації, наведені в табл. 4.2.

Таблиця 4.2 - Науково-технічні рівні та комерційні потенціали розробки

Середньоарифметична сума балів СБ, розрахована на основі висновків експертів	Науково-технічний рівень та комерційний потенціал розробки
41…48	Високий
31…40	Вищий середнього
21…30	Середній
11…20	Нижче середнього
0…10	Низький

Досягнутий науково-технічний потенціал дорівнює середньому рівню за рахунок розширення функціональних можливостей нової науково-технічної розробки порівняно з аналогами існуючими на ринку.

Покращений функціонал досягається за рахунок додавання в розробку сучасних методів машинного навчання, що надає перевагу над іншими системами виявлення аномалій в мережевих подіях. Аналіз великого обсягу даних проходить краще та з більш гнучкими можливостями, це дає розробнику можливість більш точно визначати аномальні мережеві події в SIEM системах.

Для проведення оцінки конкурентоспроможності шляхом порівняння з аналогами потрібно визначити показники за якими буде проводитись порівняння. Так технічними показниками будуть точність моделі та швидкість обробки запитів. З економічних показників це лише ціна та вартість оновлення змін за певний період часу (2 роки).

Для обчислення одиничних параметричних індексів застосовується формула, як показано на приклад оцінки індексу зміни значення параметра «Проведення аудиту»:

$$q_{зд} = \frac{P_{баз\ зд}}{P_{зд}} = \frac{92}{85,5} = 1,07, \quad (4.2)$$

Таблиця 4.3 - Оцінки якісних показників

Параметр	Одиниця виміру	Аналог	Нова розробка	Індекс зміни значення параметра	Коефіцієнт вагомості
Технічні					
Точність моделі	%	85,5	92	1,07	0,3
Швидкість обробки запитів	Ітерацій/с	125	230	1,84	0,7
Економічні					
Ціна	грн	250000	200000	1,25	0,4
Вартість впровадження змін (рік)	грн	50000	30000	0,6	0,6

Для визначення групового параметричного індексу за технічними показниками конкурентоспроможності застосуємо наступну формулу:

$$I_{ТП} = \sum_{i=1}^{4} q_i * \alpha_i = 1.07 * 0.3 + 1,84 * 0.7 = 1,6 \quad (4.3)$$

Для визначення групового параметричного індексу за економічними показниками конкурентоспроможності застосуємо наступну формулу:

$$I_{ЕП} = \sum_{i=1}^{2} q_i * \beta_i = 1,25 * 0,4 + 0,6 * 0,6 = 0,86 \quad (4.4)$$

Інтегральний показник конкурентоспроможності обчислюється за формулою:

$$К_{ІНТ} = I_{НП} * \frac{I_{ТП}}{I_{ЕП}} = 1 * \frac{1.6}{0,86} = 1.86 \quad (4.5)$$

Отриманий інтегральний показник $К_{ІНТ} = 1.86$ свідчить про перспективність конкурентоспроможності даної розробки.

З огляду на оцінки експертів та оцінку конкурентоспроможності розробки, можна зробити висновок про достатньо науково-технічний рівень, комерційний потенціал та перспективу конкурентоспроможності. В такому випадку можна зробити висновок про потенційну користь від науково-технічної розробки.

4.2 Розрахунок витрат на впровадження МКР на тему «Метод та засіб аналізу мережевих подій для SIEM-системи»

Витрати, пов’язані з виконанням науково-дослідної роботи на тему «Метод та засіб аналізу мережевих подій для SIEM-системи», під час планування, обліку та розрахунку собівартості науково-дослідної роботи класифікуються за відповідними частинами.

До частини "Витрати на оплату праці" належать витрати, пов’язані з виплатою основної та додаткової заробітної плати різним категоріям працівників. Ці витрати включають оплату праці керівників підрозділів (відділів, лабораторій, секторів, груп), а також наукових, інженерно-технічних працівників, лаборантів, робітників, студентів, аспірантів та інших працівників, які беруть безпосередню участь у виконанні конкретної теми.

Витрати на основну заробітну плату дослідників (Зо) розраховуємо у вигляді такої формули:

$$З_{\text{р}} = \sum_{i=1}^{n} \frac{M_{ni} * t_i}{T_p} \qquad (4.6)$$

де k – кількість посад дослідників, залучених до процесу досліджень;

M_{ni} – місячний посадовий оклад конкретного дослідника, грн;

t_i– кількість днів роботи конкретного дослідника, дн.;

T_p – середня кількість робочих днів в місяці, T_p=22.

$$З_o = 18000 \cdot 22 / 22 = 18000 \text{ грн.}$$

Проведені розрахунки зведемо до таблиці.

Таблиця 4.4 – Витрати на заробітну плату дослідників

Найменування посади	Місячний посадовий оклад, грн	Оплата за робочий день, грн	Кількість днів роботи	Витрати на заробітну плату, грн.
Мережевий адміністратор	18000	818,18	22	18000,00
Науковий співробітник	15000	681,81	12	8181,72
Інженер програміст	16670	757,72	15	11365,80
Оператор системи аналізу	10000	454,54	22	10000,00
Загальні витрати				47547,52

Погодинну тарифну ставку робітника відповідного розряду C_i можна визначити за формулою:

$$\mathrm{C_i} = \frac{\mathrm{M_м} * \mathrm{K_i} * \mathrm{K_c}}{\mathrm{T_р} * t_{\mathrm{зм}}} \qquad (4.7)$$

де M_M – розмір прожиткового мінімуму працездатної особи, або мінімальної місячної заробітної плати (в залежності від діючого законодавства), приймемо M_M =8000,00 грн;

K_i – коефіцієнт міжкваліфікаційного співвідношення для встановлення тарифної ставки робітнику відповідного розряду ;

K_c – мінімальний коефіцієнт співвідношень місячних тарифних ставок робітників першого розряду з нормальними умовами праці виробничих об'єднань і підприємств до законодавчо встановленого розміру мінімальної

заробітної плати.

T_p – середнє число робочих днів в місяці, приблизно T_p = 22 дн;

$t_{зм}$ – тривалість зміни, год.

$$C_1 = 8000{,}00 \cdot 1{,}35 \cdot 1{,}65 / (22 \cdot 8) = 82{,}50 \text{ грн.}$$
$$З_{р1} = 82{,}50 \cdot 16 = 1320 \text{ грн.}$$

Таблиця 4.5 – Величина витрат на основну заробітну плату робітників

Найменування робіт	Тривалість робіт, год	Розряд роботи	Тарифний коефіцієнт	Погодинна ставка, грн	Величина оплати на робітника, грн
Підключення та налаштування системи	8	3	1.35	101,25	810
Налаштування мережі	5	4	1.5	112,5	562,5
Збір початкових даних системи	16	2	1.1	82,5	1320
Розробка індивідуального ПЗ	8	4	1.5	112,5	900
Загальні витрати					3592,5

Додаткову заробітну плату розраховуємо як 10 … 12% від суми основної заробітної плати дослідників та робітників за формулою:

$$З_{дод} = (З_о + З_р) * \frac{Н_{дод}}{100\%} = (47547{,}52 + 3592{,}5) \cdot 10 / 100\% = 5114{,}00, \quad (4.8)$$

Нарахування на заробітну плату дослідників та робітників розраховуємо як 22% від суми основної та додаткової заробітної плати дослідників і робітників за формулою:

$$З_н = (З_о + З_р + З_{дод}) * \frac{Н_{зп}}{100\%} = (47547{,}52 + 3592{,}5 + 5114{,}00) \cdot \frac{22}{100}\% = 12375{,}88 \quad (4.9)$$

Майже всі процеси відбуваються дистанційно і через мережу, але є деяка необхідність у папері для записів, створення моделей та друку.

Витрати на матеріали (*М*), у вартісному вираженні розраховуються окремо по кожному виду матеріалів за формулою:

$$М1 = 2{,}0 \cdot 172{,}00 \cdot 1{,}1 = 378{,}4 \text{ грн.}, \quad (4.10)$$

де H_j – норма витрат матеріалу *j*-го найменування, кг;

n – кількість видів матеріалів;

$Ц_j$ – вартість матеріалу *j*-го найменування, грн/кг;

$К_j$ – коефіцієнт транспортних витрат, ($К_j$ = *1,1 … 1,15*);

B_j – маса відходів *j*-го найменування, кг;

$Ц_{вj}$ – вартість відходів *j*-го найменування, грн/кг.

Таблиця 4.6 – Витрати на матеріали

Найменування матеріалу, марка,тип, сорт	Ціна за 1 кг, грн	Норма витрат, кг	Вартість витраченого матеріалу, грн
Папір канцелярський	172,00	2,0	378,40
Папір для заміток	118,20	2,0	260,04
Канцелярські прибори (ручки, олівці)	157,50	0,5	86,62
Картридж для принтеру	550	1 шт	605,00
Флеш память USB Kingston DataTraveler Exodia Onyx 64GB USB 3.2 Gen 1	200	2 шт	440,00
Всього			1770,06

Витрати за частиною «Службові відрядження» розраховуємо як 20…25% від суми основної заробітної плати дослідників та робітників за формулою:

$B_{св} = (З_о + З_р) * \frac{Н_{св}}{100\%} = (47547{,}52 + 3592{,}5) \cdot 20 / 100\% = 10228{,}00$ *грн.,* (4.11)

Витрати за статтею «Інші витрати» розраховуємо як 50…100% від суми основної заробітної плати дослідників та робітників за формулою:

$I_в = (З_о + З_р) * \frac{Н_{св}}{100\%} = (47547{,}52 + 3592{,}5) \cdot 50 / 100\% = 25570{,}00$ *грн ,* (4.12)

Розрахунок витрат на комплектуючі які використовують при дослідженні нового технічного рішення розраховуються за наступною формулою:

$$K_в = \sum_{i=1}^{n} H_j * Ц_j * K_j = 2 * 25715 * 1.1 = 56573$$

де Нj – кількість комплектуючих j-го виду, шт.;

Цj – покупна ціна комплектуючих j-го виду, грн;

Кj – коефіцієнт транспортних витрат, (Кj = 1,1 … 1,15).

Таблиця 4.7 - Витрати на комплектуючі

Найменування комплектуючих	Кількість, шт.	Ціна за штуку, грн	Вартість, грн
Сервер DELL T620 (16x2.5) SFF	2	25 715	56573
Накопичувач SSD Intel [800GB SATA 2.5" S3700] (SSDSC2BA800G3)	2	5072	11158.4
Маршрутизатор MikroTik RB4011iGS+5HacQ2HnD-IN	1	10699	11768.9
Кабель LAN CAT 5E(10м)	10	153	1683
МФУ Canon Pixma TS3340, Wi Fi (3771C007AA / 3771C007BA)	1	4 199	4618
Всього			85802,2

Витрати на силову електроенергію розраховуємо за формулою:

$$B_e = \sum_{i=1}^{n} \frac{M_{yi} \cdot t_i \cdot Ц_е \cdot К_{впі}}{\eta_i}, \quad (4.13)$$

де Wyi – встановлена потужність обладнання на визначеному етапі розробки, кВт;

ti – тривалість роботи обладнання на етапі дослідження, год;

Це – вартість 1 кВт-години електроенергії, грн; (вартість електроенергії визначається за даними енергопостачальної компанії), Це = 10,5 грн;

Квпі – коефіцієнт, що враховує використання потужності, Квпі <1

ηі – коефіцієнт корисної дії обладнання, ηі<1.

$$B_e = 0.75 * 168 * 10{,}5 * \frac{0{,}95}{0{,}9} = 1396{,}5$$

Таблиця 4.8 - Витрати на електроенергію

Найменування обладнання	Встановлена потужність, кВт	Тривалість роботи, год	Сума, грн
Сервер DELL T620 (16x2.5) SFF	0,75	168	1396,5
Маршрутизатор MikroTik RB4011iGS+5HacQ2HnD-IN	0.044	168	81,92
Принтер Canon Pixma TS3340, Wi Fi (3771C007AA / 3771C007BA)	0.011	10	1,21
Всього			1479,63

До частини «Накладні (загальновиробничі) витрати включають: витрати, пов’язані з управлінням організацією; витрати на винахідництво та раціоналізацію; витрати на підготовку (перепідготовку) та навчання кадрів; витрати, пов’язані з набором робочої сили; витрати на оплату послуг банків; витрати, пов’язані з освоєнням виробництва продукції; витрати на науково технічну інформацію та рекламу та ін.

$$В_{нзв} = З_0 + З_р + З_{дод} + B_e + З_н + В_{прг} + А_{обл} + М + B_e + В_{нзв}, \quad (4.14)$$

$$B_{заг} = 47547{,}52 + 3592{,}5 + 1770{,}06 + 10228{,}00 + 25570{,}00 + 85802{,}2 + 1479{,}63 = 175989.91$$

Загальні витрати *ЗВ* на завершення науково-дослідної (науково-технічної) роботи та оформлення її результатів розраховується за формулою:

$$ЗВ = \frac{В_{заг}}{\eta} = 175989.91/0{,}9 = 195544{,}34, \quad (4.15)$$

4.3 Розрахунок економічної ефективності науково-технічної розробки за її можливої комерціалізації потенційним інвестором

Для оцінки економічної доцільності виведення цієї науково-технічної розробки на ринок слід врахувати такі аспекти, як тривалість впровадження, прогнозований термін отримання прибутку для інвесторів, рівень попиту та його основні споживачі, а також ринкову ціну аналогічних рішень із подібними можливостями.

Для прикладу наведемо обчислення прибутку інвестора в першому році після реалізації:

$$\Delta П_1 = (\pm \Delta Ц_0 * N + Ц_0 * \Delta N)_i * \lambda * \rho * \left(1 - \frac{\vartheta}{100}\right)$$
$$= (200000 * 4 + 150000 * 0) * 0{,}8333 * 0{,}33 * (1 - 0{,}18) = 214216{,}43$$

Другому році:

$$\Delta П_2 = (200000 * 4 + 200000 * 4) * 0{,}8333 * 0{,}33 * (1 - 0{,}18) = 360785{,}56$$

Третьому році:

$$\Delta П_3 = (200000 * 4 + 200000 * 8) * 0{,}8333 * 0{,}33 * (1 - 0{,}18) = 541178{,}35$$

Далі необхідно розрахувати приведену вартість всіх чистих прибутків ПП за формулою:

$$ПП = \sum_{i=1}^{T} \frac{\Delta П_i}{(1+\tau)^t} = \frac{214216{,}43}{1{,}15^1} + \frac{360785{,}56}{1{,}15^2} + \frac{541178{,}35}{1{,}15^3} =$$

$$186275{,}15 + 272805{,}71 + 355833{,}54 = 814914.4$$

Необхідно розрахувати також початкові інвестиції PV. Обчислення проводиться за формулою:

$$PV = k_{розр} * ЗВ = 2 * 195544{,}34 = 391088{,}68$$

Також необхідно обчислити приведений дохід за формулою:

$$E_{абс} = ПП - PV = 814914{,}4 \ - 391088{,}68 = \ 423825{,}72$$

Для остаточного прийняття рішення з цього питання необхідно розрахувати внутрішню економічну дохідність $E_в$ або показник внутрішньої норми дохідності (IRR, Internal Rate of Return) вкладених інвестицій за формулою:

$$E_{\text{в}} = \sqrt[\text{Тж}]{1 + \frac{\text{Е}_{\text{абс}}}{PV}} - 1 = \sqrt[2]{1 + \frac{423825{,}72}{391088{,}68}} - 1 = 0.44$$

Для визначення бар'єрної ставки, з якою необхідно порівняти внутрішню економічну дохідність, використовується формула:

$$\tau_{\text{мін}} = d + f = 0.15 + 0.3 = 0.35$$

Оскільки внутрішня економічна дохідність перевищує мінімальну, потенційний інвестор можу бути зацікавлений в даній розробці.

Також обчислимо період окупності інвестицій за формулою:

$$\text{Т}_{\text{ок}} = \frac{1}{E_{\text{в}}} = \frac{1}{0.44} = 2{,}27$$

Оскільки $\text{Т}_{\text{ок}} < 3$ років, можна зробити висновок, що впровадження науково-технічної розробки є економічно ефективним.

Науково-технічна розробка має високий інвестиційний потенціал завдяки значним очікуваним доходам та прийнятним термінам окупності. Для детального аналізу економічної ефективності було проведено оцінку її науково-технічного рівня та комерційного потенціалу трьома експертами за встановленими критеріями. Середній бал за результатами оцінки склав 29,33 , що свідчить про високий рівень науково-технічного розвитку та комерційної перспективності.

Додатково було виконано аналіз конкурентоспроможності, який дозволив визначити ключові переваги технічних характеристик і економічних показників розробки. Результати цього аналізу підтвердили, що розробка є конкурентоспроможною та готовою до виходу на ринок.

Також здійснено розрахунки витрат на проведення науково-дослідних робіт і оцінку економічної ефективності впровадження розробки на підприємстві. Результати підтвердили, що її впровадження є економічно вигідним.

ВИСНОВКИ

В магістерській кваліфікаційній роботі було проведено розробку моделі та засобу аналізу мережевих подій для SIEM систем. Покращення методу аналізу дозволило підвищити рівень безпеки мережі та підвищити якість роботи моделі.

В першому розділі було проаналізовано існуючі принципи роботи SIEM систем, розглянуто наявні засоби на ринку та проведено їх порівняння для виявлення сильних сторін на які потрібно було орієнтуватись при розробці та слабких сторін, яких варто було уникати. Було також проведено аналіз існуючих технологій машинного навчання, що дало змогу обрати підходящі алгоритми для їх покращення та застосування в роботі

В другому розділі роботи було розроблено метод аналізу мережевих подій, який полягає в застосуванні гібридного підходу для аналізу як категоризованих подій, так і нових і невідомих. На основі цього методу було створено декілька алгоритмів які базуються на існуючих з модифікаціями призначеними для більш детальної роботи та аналізу.

В третьому розділі було розроблено засіб аналізу мережевих подій у вигляді програмного застосунку на платформі Windows. Роботу головних частин програмного коду було описано та наведено пояснення щодо їх застосування. Було проведено тестування роботи засобу з різними вхідними даними, при різних налаштуваннях та продемонстровано у вигляді візуалізованих графіків.

В четвертому розділі роботи було проведено дослідження комерційної та технічної цінності роботи, проведено розрахунки вартості дослідження та впровадження, а також проведено розрахунок можливого прибутку за залучення інвесторів.

В результаті було отримано метод та засіб аналізу мережевих подій для SIEM систем який відповідає всім поставленим вимогам і задачам, а також в певних аспектах є кращим за свої ринкові аналоги.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1.Pareek R. Network Security: An Approach Towards Secure Computing. [Електронийресурс].URL: https://www.researchgate.net/publication/228442094_Network_Security_An_Approach_Towards_Secure_Computing.

2. Information Security Management Principles / D. Sutton et al. BCS Learning & Development Limited, 2020. 268 p.

3. Theyazn H.H Aldhyani. A review of network traffic analysis and prediction techniques [Електронний ресурс]. - Режим доступу: https://www.researchgate.net/publication/339927785_A_review_of_network_traffic_analysis_and_prediction_techniques

4. Information Security Management Principles / D. Sutton et al. BCS Learning & Development Limited, 2020. 268 p.

5. Грайворонський М., Новіков О. Безпека інформаційно-комунікаційних систем. Київ : Вид. група BHV, 2009. 608 с.

6. Davidoff S. Network forensics: Tracking hackers through cyberspace. Upper Saddle River, NJ : Prentice Hall, 2012. 576 p.

7. Miller D. Security information and event management (SIEM) implementation. New York : McGraw-Hill, 2011. 430 p.

8. Sanders C., Smith J. Applied Network Security Monitoring: Collection, Detection, and Analysis. Elsevier Science & Technology Books, 2013. 496 p.

9. Hubballi N., Suryanarayanan V. False alarm minimization techniques in signature-based intrusion detection systems: A survey. [Електронний ресурс]. URL: https://www.sciencedirect.com/science/article/abs/pii/S0140366414001480?via=ihub.

10. Aljawarneh S., Aldwairi M., Yassein M. B. Anomaly-based intrusion detection system through feature selection analysis and building hybrid efficient model. [Електронний ресурс]. URL: https://www.sciencedirect.com/science/article/abs/pii/S1877750316305099?via=ihub.

11. A SIEM Architecture for Multidimensional Anomaly Detection / T. Laue et al. [Електронний ресурс]. URL: https://ieeexplore.ieee.org/document/9660903.

12. Integration of Splunk Enterprise SIEM for DDoS Attack Detection in IoT / M. Hristov et al. [Електронний ресурс]. URL: https://ieeexplore.ieee.org/document/9685977.

13. Anumol E. T. Use of Machine Learning Algorithms with SIEM for Attack Prediction. Advances in Intelligent Systems and Computing. 2015. P. 231–235.

14. Sarker, I.H. Machine Learning: Algorithms, Real-World Applications and Research Directions. SN COMPUT. SCI. 2, 160 (2021). https://doi.org/10.1007/s42979-021-00592-x

15. Taye M. M. Understanding of Machine Learning with Deep Learning: Architectures, Workflow, Applications and Future Directions. [Електронний ресурс]. URL: https://www.mdpi.com/2073-431X/12/5/91.

16. Ghosh S. How to Compare Machine Learning Models and Algorithms. [Електронний ресурс]. URL: https://neptune.ai/blog/how-to-compare-machine-learning-models-and-algorithms.

17. Le J. The Top 10 Machine Learning Algorithms to Know. [Електронний ресурс]. URL: https://builtin.com/data-science/tour-top-10-algorithms-machine-learning-newbies.

18. Park H-S, Jun C-H. A simple and fast algorithm for k-medoids clustering. Expert Syst Appl. 2009;36(2):3336–41

19. Захист інформації в інформаційно-комунікаційних системах: Методичні вказівки, завдання на курсовий проект /Уклад.: О. П. Войтович, Л. М. Куперштейн, В. А. Каплун. - Вінниця : ВНТУ, 2016, - 26 с. – [Електронний ресурс]. – URL: https://iq.vntu.edu.ua/method/getfile.php?fname=54033.pdf&x=1&card_id=11291&id=54033

20. Гончар Д. А. Способи аналізу мережевих подій в SIEM-системах [Електронний ресурс] / Д. А. Гончар, В. В. Лукічов // Матеріали Всеукраїнської науково-практичної інтернет-конференції "Молодь в науці: дослідження, проблеми, перспективи (МН-2023)", Вінниця, 22-23 червня 2023 р. – Електрон. текст. дані. – 2023. – Режим доступу: https://conferences.vntu.edu.ua/index.php/mn/mn2023/paper/view/18029

21. Гончар Д. А. Методи Аналізу Мережевих Подій З Використанням Машинного Навчання [Електронний ресурс] / Д. А. Гончар, В. В. Лукічов // Матеріали Всеукраїнської науково-практичної інтернет-конференції "Молодь в науці: дослідження, проблеми, перспективи (МН-2025)", Вінниця, 2024 р. – Електрон. текст. дані. – 2024. – Режим доступу: https://conferences.vntu.edu.ua/index.php/mn/mn2025/paper/view/22605

22. Jupud L. Machine learning techniques using python for data analysis in performance evaluation. [Електронний ресурс]. URL: https://www.researchgate.net/publication/324928841_Machine_learning_techniques_using_python_for_data_analysis_in_performance_evaluation.

23. Mckinney W. Pandas: a Foundational Python Library for Data Analysis and Statistics. [Електронний ресурс]. URL: https://www.researchgate.net/publication/265194455_pandas_a_Foundational_Python_Library_for_Data_Analysis_and_Statistics.

24. Fuentes F. NumPy: The Fundamental Tool for Data Science in Python. [Електронний ресурс]. URL: https://medium.com/@m.franfuentes/numpy-the-fundamental-tool-for-data-science-in-python-fa2b605a3bf9.

25. Crudu A. Exploring Python GUI Frameworks: Tkinter, PyQT, and wxPython. [Електронний ресурс]. URL: https://moldstud.com/articles/p-exploring-python-gui-frameworks-tkinter-pyqt-and-wxpython.

ДОДАТКИ

ДОДАТОК А

ПРОТОКОЛ ПЕРЕВІРКИ

НАВЧАЛЬНОЇ (КВАЛІФІКАЦІЙНОЇ) РОБОТИ

Назва роботи: Метод та засіб аналізу мережевих подій для SIEM-системи
Автор роботи: Гончар Данило Андрійович
Тип роботи: магістерська кваліфікаційна робота
Підрозділ кафедра захисту інформації ФІТКІ
Керівник Лукічов Віталій Володимирович, к.т.н., доцент

Показники звіту подібності

Turnitin	
Оригінальність	97 %
Загальна схожість	3 %

Аналіз звіту подібності (відмітити потрібне):

√ 1. Запозичення, виявлені у роботі, оформлені коректно і не містять ознак плагіату.

□ 2. Виявлені у роботі запозичення не мають ознак плагіату, але їх надмірна кількість викликає сумніви щодо цінності роботи і відсутності самостійності її автора. Роботу направити на доопрацювання.

□ 3. Виявлені у роботі запозичення є недобросовісними і мають ознаки плагіату та/або в ній містяться навмисні спотворення тексту, що вказують на спроби приховування недобросовісних запозичень.

Заявляю, що ознайомлений з повним звітом подібності, який був згенерований Системою щодо роботи.

Автор роботи _____ Гончар Д. А.
(підпис) (прізвище, ініціали)

Особа, відповідальна за перевірку _____ Каплун В. А.
(підпис) (прізвище, ініціали)

Керівник роботи _____ Лукічов В.В.
(підпис) (прізвище, ініціали)

ДОДАТОК Б. КОД ПРОГРАМИ

```
import numpy as np
import pandas as pd
import requests
from requests.auth import HTTPBasicAuth
from sklearn.metrics import pairwise_distances
import matplotlib.pyplot as plt
from sklearn.ensemble import IsolationForest
from sklearn.cluster import KMeans
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
import tkinter as tk
from tkinter import messagebox

def wazuh_connection():
    # Налаштування
    WAZUH_API = "https://<WAZUH_MANAGER_IP>"
    USER = "<USER>"
    PASSWORD = "<PASSWORD>"

    # Запит до API
    response = requests.get(
        f"{WAZUH_API}/alerts?agent_id=<AGENT_ID>",
        auth=HTTPBasicAuth(USER, PASSWORD),
        verify=False
    )

    # Перевірка результату
    if response.status_code == 200:
        print("Connection successful!")
        print(response.json())
    else:
        print(f"Error: {response.status_code}")

def k_means_visual(network_data):
    network_data = network_data.iloc[:500:][['dbytes', 'dpkts']]
    # Кластеризація K-Means
    kmeans = KMeans(n_clusters=3)
    kmeans.fit(network_data)
    labels = kmeans.labels_
    centroids = kmeans.cluster_centers_

    # Обчислення відстаней до центроїдів
    distances = np.linalg.norm(network_data - centroids[labels], axis=1)

    # Визначення порогового значення (персентиль)
    thresholds = {}
    for i in range(len(centroids)):
        cluster_distances = distances[labels == i]
```

```
            threshold = np.percentile(cluster_distances, 80)  # 90-й персентиль
            thresholds[i] = threshold

        # Визначення аномальних точок
        anomalies = []
        for i, distance in enumerate(distances):
            cluster_id = labels[i]
            if distance > thresholds[cluster_id]:
                anomalies.append(i)

        anomalies = np.array(anomalies)  # Масив індексів аномальних точок

        # Візуалізація
        # Координати аномальних точок
        anomalies_2d = network_data.iloc[anomalies]

        # Побудова графіка
        plt.scatter(network_data.iloc[:, 0], network_data.iloc[:, 1], c='blue',
label='Нормальні точки')
        plt.scatter(anomalies_2d.iloc[:, 0], anomalies_2d.iloc[:, 1], c='red',
label='Аномальні точки')
        plt.scatter(centroids[:, 0], centroids[:, 1], c='green', marker='x',
label='Центроїди')

        # Додати лінії між точками та центроїдами
        for i in range(len(network_data)):
            # З'єднуємо точку з її центроїдом
            plt.plot([network_data.iloc[i, 0], centroids[labels[i], 0]],
                     [network_data.iloc[i, 1], centroids[labels[i], 1]],
                     c='gray', linestyle='dashed', alpha=0.5)

        plt.xlabel('Розмір переданих даних (байти)')
        plt.ylabel('Кількість пакетів')
        plt.legend()
        plt.grid(True)
        plt.title('Візуалізація кластерів з аномаліями та лініями до центроїдів')

        plt.show()

    def isolation_forest_labels(network_data_scaled, label_column="label"):
        # Вибираємо перші 500 записів і потрібні колонки
        network_data_scaled = network_data_scaled.iloc[:500][['dur', 'dinpkt',
'label']]
        print(network_data_scaled.head(10))

        # Розділення на ознаки X та мітки y
        X = network_data_scaled.drop(columns=[label_column]).values  # Перетворюємо
на NumPy-масив
        y = network_data_scaled[label_column].values  # Мітки аномалій (0 - нормальні,
1 - аномальні)
```

```
        # Розділяємо дані на тренувальний та тестовий набори
        X_train, X_test, y_train, y_test = train_test_split(X, y, stratify=y, random_state=42)

        # Побудова Isolation Forest
        clf = IsolationForest(max_samples=100, random_state=0)
        clf.fit(X_train)

        # Визначення аномальних точок
        y_pred = clf.predict(X_test)
        anomalies = y_pred == -1

        # Візуалізація результатів Isolation Forest
        # Використовуємо лише дві ознаки для графіку
        plt.scatter(X_test[:, 0], X_test[:, 1], c=anomalies, s=20, edgecolor="k")
        plt.title("Результати Isolation Forest")
        plt.xlabel("Тривалість з'єднання (dur)")
        plt.ylabel("Інтервал між пакетами (dinpkt)")
        plt.legend(["Нормальні", "Аномальні"])
        plt.grid(True)
        plt.show()


    def file_processing():
        # Завантаження даних
        file_path = "data_csv/UNSW_NB15_testing-set.csv"
        data = pd.read_csv(file_path)
        print("Number of rows:", data.shape[0])

        # Крок 1: Вибір ознак
        selected_columns = [
            "dur", "spkts", "dpkts", "sbytes", "dbytes", "rate",
            "sload", "dload", "sttl", "dttl", "sinpkt", "dinpkt",
            "tcprtt", "synack", "ackdat", "proto", "service", "state", "label"
        ]
        data_selected = data[selected_columns]

        # Крок 2: Закодування категорійних змінних
        categorical_columns = ["proto", "service", "state"]
        label_encoders = {}
        for column in categorical_columns:
            le = LabelEncoder()
            data_selected[column] = le.fit_transform(data_selected[column].astype(str))
            label_encoders[column] = le  # Збереження кодера для подальшого використання

        # Крок 3: Масштабування числових ознак
        numerical_columns = [col for col in selected_columns if col not in categorical_columns + ["label"]]
        scaler = StandardScaler()
```

```
        data_selected[numerical_columns] = scaler.fit_transform(data_selected[numerical_columns])

        # Перевірка підготовлених даних
        print("Перші кілька рядків підготовлених даних:")
        print(data_selected.head())

        # Підготовка до тренування
        X = data_selected.drop(columns=["label"])  # Дані для тренування
        y = data_selected["label"]  # Мітки

        # Збереження підготовлених даних у файл (за бажанням)
        output_path = "preprocessed_data.csv"
        data_selected.to_csv(output_path, index=False)
        print(f"Підготовлені дані збережено у файл: {output_path}")
    def file_loader():
        file_path = "data_csv/UNSW_NB15_training-set.csv"
        data = pd.read_csv(file_path)
        return data


    class AdaptiveLogisticRegression:
        def __init__(self, learning_rate=0.01, iterations=1000, lambda_reg=0.01):
            self.learning_rate = learning_rate
            self.iterations = iterations
            self.lambda_reg = lambda_reg

            # Initialize lists for storing accuracy at each iteration
            self.train_accuracy = []
            self.test_accuracy = []

        def fit(self, X, y):
            # Scale the data
            scaler = StandardScaler()
            X_scaled = scaler.fit_transform(X)

            # Initialize weights and adaptive weights
            self.weights = np.zeros(X_scaled.shape[1])
            self.adaptive_weights = np.ones(X_scaled.shape[1])  # Initialize adaptive weights

            # Split the data into training and test sets
            X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.3, random_state=42)

            # Train the model
            for i in range(self.iterations):
                # Make predictions
                predictions = self.predict_proba(X_train)

                # Calculate the error
```

```
            error = predictions - y_train

            # Update weights with adaptive step size
            gradient = X_train.T.dot(error) / len(y_train) + self.lambda_reg *
self.weights
            self.weights -= self.learning_rate * gradient /
(self.adaptive_weights + 1e-6)

            # Update adaptive weights based on errors
            self.adaptive_weights += gradient ** 2

            # Calculate accuracy on training and test sets
            train_pred = self.predict(X_train)
            test_pred = self.predict(X_test)
            self.train_accuracy.append(accuracy_score(y_train, train_pred))
            self.test_accuracy.append(accuracy_score(y_test, test_pred))

            if i % 100 == 0:  # For progress tracking
                print(f"Iteration {i}: Loss = {self.compute_loss(X_train,
y_train)}")

    def predict_proba(self, X):
        # Logistic activation function
        return 1 / (1 + np.exp(-X.dot(self.weights)))

    def predict(self, X):
        # Convert probabilities to class labels (0 or 1)
        return (self.predict_proba(X) >= 0.5).astype(int)

    def compute_loss(self, X, y):
        # Compute logistic loss with regularization
        predictions = self.predict_proba(X)
        loss = -np.mean(y * np.log(predictions + 1e-6) + (1 - y) * np.log(1 -
predictions + 1e-6))
        regularization = self.lambda_reg * np.sum(self.weights ** 2) / 2
        return loss + regularization

def test_regression(network_data, label_column ="label"):
    network_data = network_data.iloc[:20000][['dur', 'dinpkt', 'label']]
    # Генерація набору даних
    X = network_data.drop(columns=[label_column]).values  # Перетворюємо на
NumPy-масив
    y = network_data[label_column].values  # Мітки аномалій (0 - нормальні, 1 -
аномальні)

    # Розділення на тренувальний та тестовий набори
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,
random_state=42)

    # Ініціалізація та тренування моделі
    model = AdaptiveLogisticRegression(learning_rate=0.01, iterations=1000,
```

```
lambda_reg=0.1)
        model.fit(X_train, y_train)
        print(model.train_accuracy*100)
        plt.plot(range(model.iterations), model.train_accuracy, label='Тренувальна
точність')
        plt.plot(range(model.iterations), model.test_accuracy, label='Тестова
точність')
        plt.xlabel('Ітерації')
        plt.ylabel('Точність')
        plt.title('Адаптивна логістична регресія')
        plt.legend()
        plt.show()


    class WeightedKNN:
        def __init__(self, k=5):
            self.k = k

        def fit(self, X_train, y_train):
            # Зберігаємо навчальні дані
            self.X_train = X_train
            self.y_train = y_train

        def predict(self, X_test):
            # Прогнозування класів для тестових даних
            predictions = []
            for i, x_test in enumerate(X_test):
                # Обчислюємо відстань до всіх точок в навчальному наборі
                distances = pairwise_distances([x_test], self.X_train,
metric='euclidean').flatten()
                # Отримуємо індекси найближчих сусідів
                nearest_indices = np.argsort(distances)[:self.k]
                # Отримуємо відповідні класи для найближчих сусідів
                nearest_classes = self.y_train[nearest_indices]
                # Обчислюємо ваги сусідів на основі відстаней
                weights = 1 / (distances[nearest_indices] + 1e-6)  # Додаємо мале
значення, щоб уникнути ділення на 0
                # Вибираємо клас, який має найбільшу суму ваг
                weighted_class = np.bincount(nearest_classes,
weights=weights).argmax()
                predictions.append(weighted_class)
            return np.array(predictions)

    def test_knn(network_data, label_column = "label"):
        network_data = network_data.iloc[:50000][['dur', 'dinpkt', 'label']]
        # Генерація набору даних
        X = network_data.drop(columns=[label_column]).values  # Перетворюємо на
NumPy-масив
        y = network_data[label_column].values  # Мітки аномалій (0 - нормальні, 1 -
аномальні)
```

```
        # Розбиття на навчальні та тестові набори
        X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,
random_state=42)

        # Ініціалізація і тренування моделі
        knn_model = WeightedKNN(k=5)
        knn_model.fit(X_train, y_train)

        # Прогнозування на тестовому наборі
        y_pred = knn_model.predict(X_test)

        # Оцінка точності
        accuracy = accuracy_score(y_test, y_pred)
        print(f"Точність моделі: {accuracy * 100:.2f}%")

        # Візуалізація точності
        plt.hist([y_test, y_pred], bins=2, color=['blue', 'red'], label=["True",
"Predicted"])
        plt.legend()
        plt.title(f'Справжні vs Прогнозовані значення (Точність: {accuracy *
100:.2f}%)')
        plt.show()

    # Функції-обгортки для виконання вибраних дій
    def run_test_knn():
        try:
            test_knn(file_loader())
            messagebox.showinfo("Успіх", "Класифікація виконана успішно.")
        except Exception as e:
            messagebox.showerror("Помилка", f"Виникла помилка: {e}")

    def run_test_regression():
        try:
            test_regression(file_loader())
            messagebox.showinfo("Успіх", "Логістична регресія виконана успішно.")
        except Exception as e:
            messagebox.showerror("Помилка", f"Виникла помилка: {e}")

    def run_isolation_forest():
        try:
            isolation_forest_labels(file_loader())
            messagebox.showinfo("Успіх", "Аналіз аномальних відхилень виконано
успішно.")
        except Exception as e:
            messagebox.showerror("Помилка", f"Виникла помилка: {e}")

    def run_k_means():
        try:
            k_means_visual(file_loader())
            messagebox.showinfo("Успіх", "Кластеризація виконана успішно.")
        except Exception as e:
```

```
            messagebox.showerror("Помилка", f"Виникла помилка: {e}")

    def run_all_functions():
        try:
            test_knn()
            test_regression()
            isolation_forest_labels(file_loader())
            k_means_visual(file_loader())
            messagebox.showinfo("Успіх", "Усі функції виконані успішно.")
        except Exception as e:
            messagebox.showerror("Помилка", f"Виникла помилка: {e}")

    # Створення головного вікна
    root = tk.Tk()
    root.title("Меню вибору функцій")

    # Опис інтерфейсу
    label = tk.Label(root, text="Виберіть функцію для виконання:", font=("Arial",
14))
    label.pack(pady=10)

    btn_k_means = tk.Button(root, text="Виконати кластеризацію з аналізом",
font=("Arial", 12), command=run_k_means)
    btn_k_means.pack(pady=5)
    btn_isolation_forest = tk.Button(root, text="Виконати виявлення аномальних
відхилень", font=("Arial", 12), command=run_isolation_forest)
    btn_isolation_forest.pack(pady=5)
    btn_test_regression = tk.Button(root, text="Виконати логістичну регресію",
font=("Arial", 12), command=run_test_regression)
    btn_test_regression.pack(pady=5)
    btn_test_knn = tk.Button(root, text="Виконати класифікацію точок", font=("Arial",
12), command=run_test_knn)
    btn_test_knn.pack(pady=5)
    btn_all = tk.Button(root, text="Виконати всі функції послідовно", font=("Arial",
12), command=run_all_functions)
    btn_all.pack(pady=10)
    root.mainloop()
```

ІЛЮСТРАТИВНА ЧАСТИНА

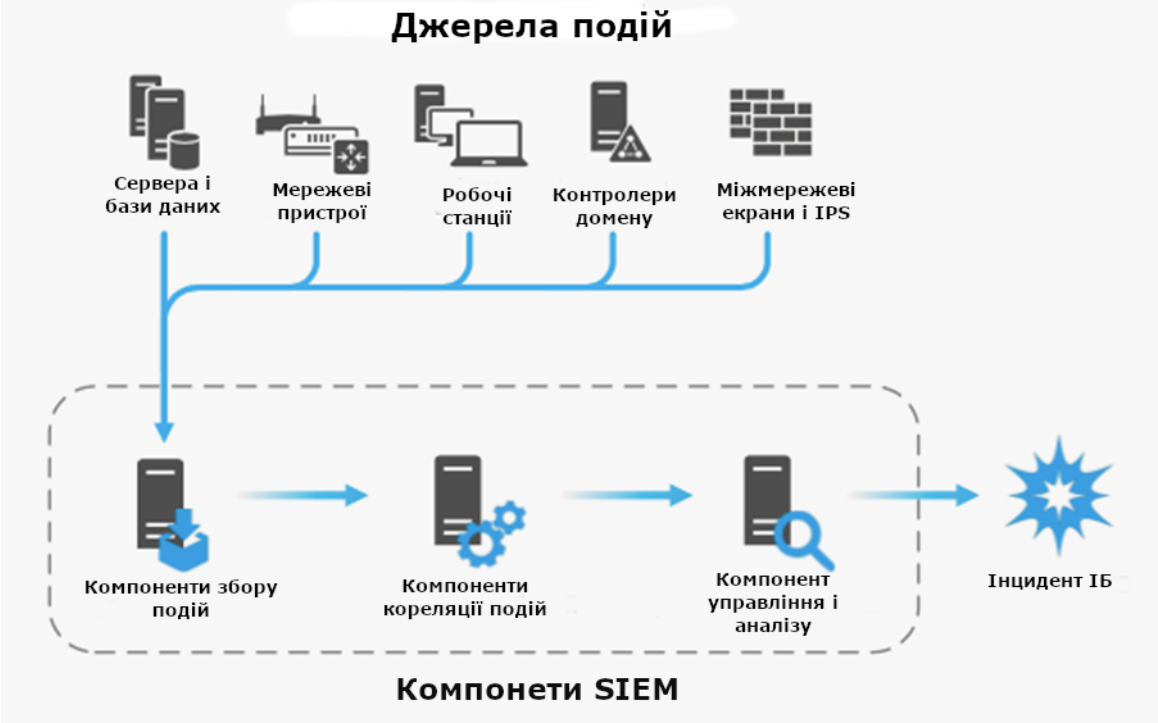
МЕТОД ТА ЗАСІБ АНАЛІЗУ МЕРЕЖЕВИХ ПОДІЙ ДЛЯ SIEM СИСТЕМИ

(Назва бакалаврської кваліфікаційної роботи)

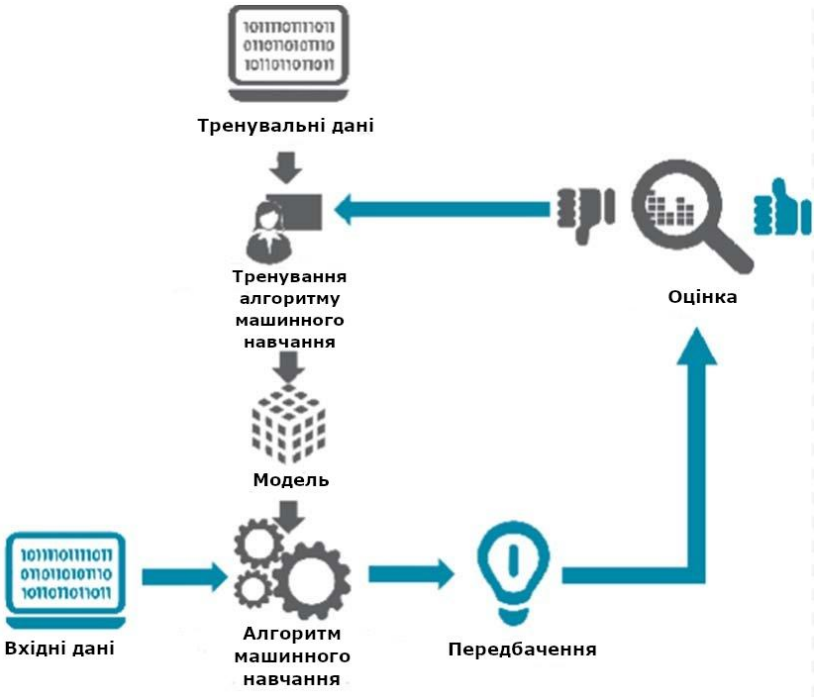
Виконав: студент 1БС-23м
спеціальності 125 Кібербезпека та захист інформації
_____ Данило ГОНЧАР
_13.12_____ 2024 р.

Керівник: к. т. н., доцент каф. ЗІ
_____Віталій ЛУКІЧОВ
_13.12_____ 2024 р.

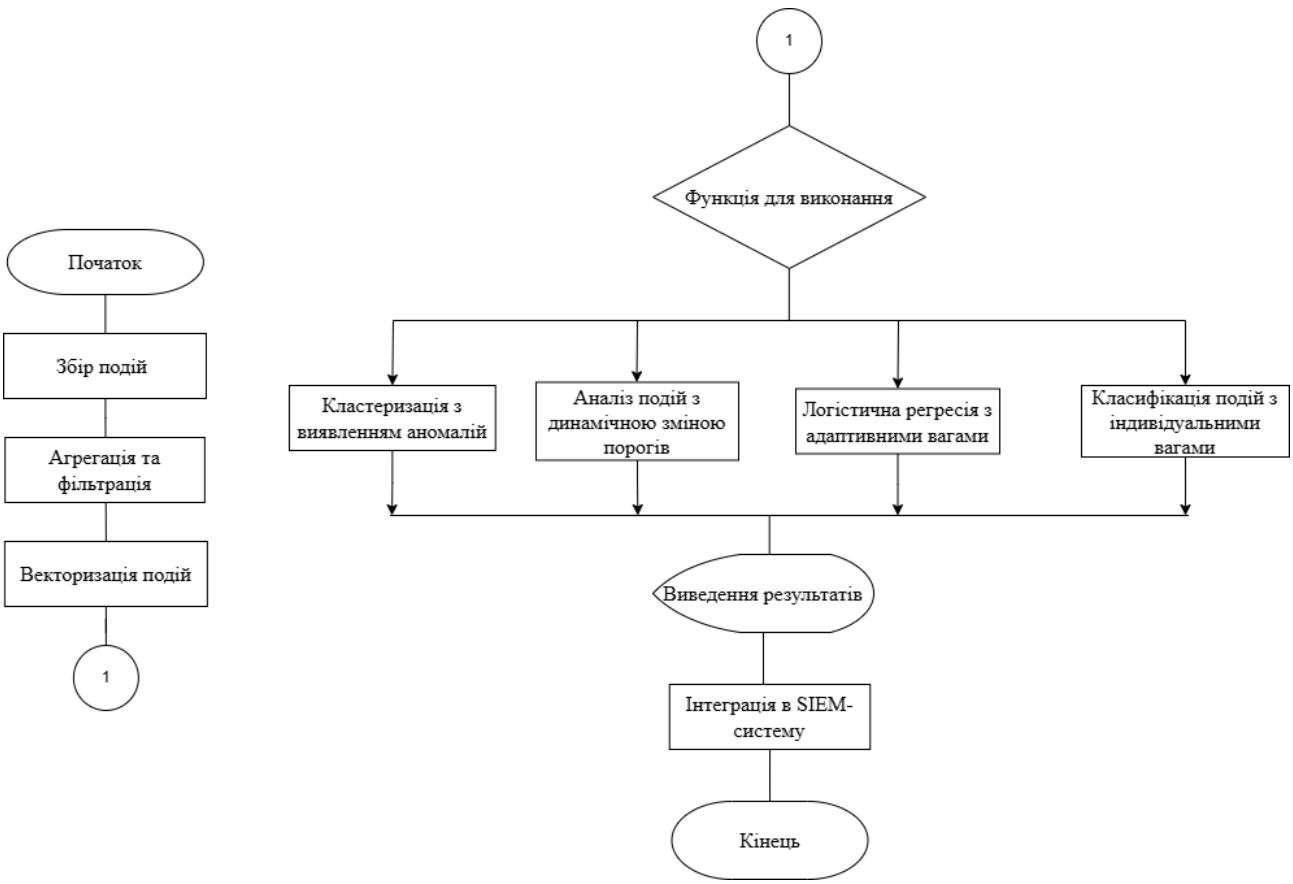
СХЕМА РОБОТИ SIEM СИСТЕМ



ПІДХІД МАШИННОГО НАВЧАННЯ

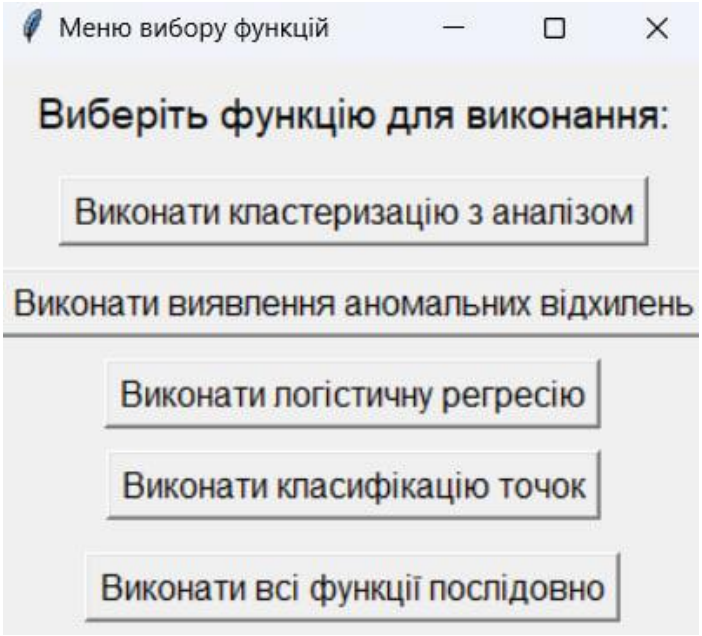


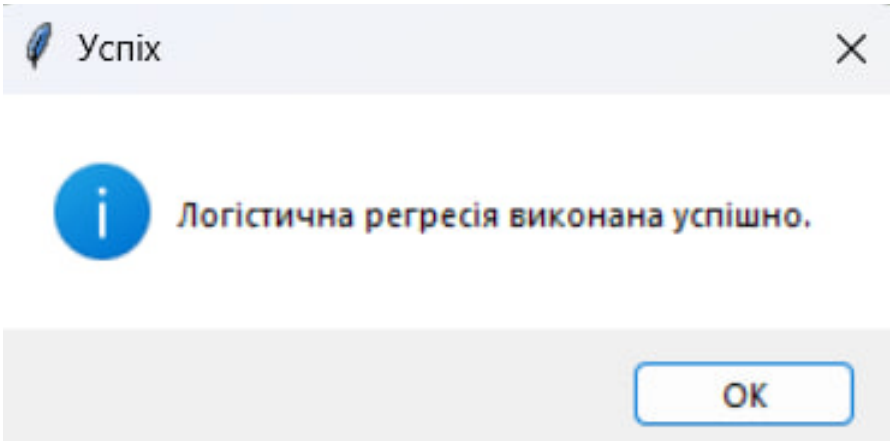
ЗАГАЛЬНА СХЕМА РОБОТИ ЗАСОБУ

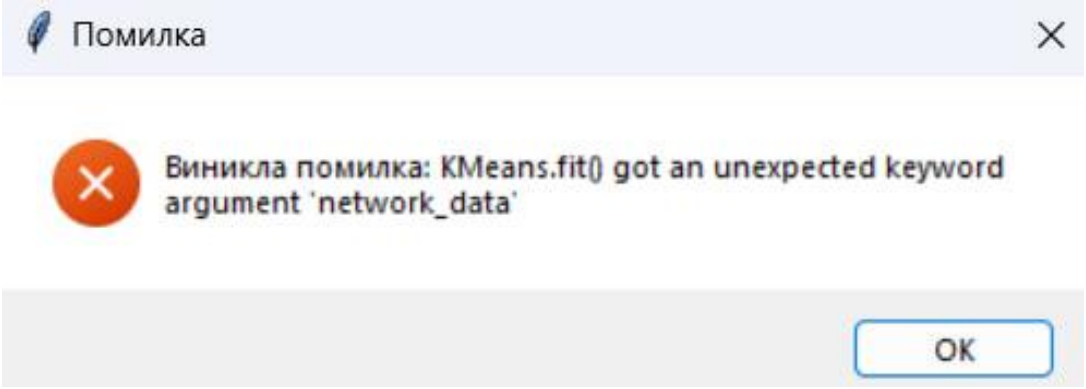


СХЕМИ РОБОТИ З SIEM СИСТЕМОЮ

ГРАФІЧНИЙ ІНТЕРФЕЙС КОРИСТУВАЧА ЗАСОБУ







РЕЗУЛЬТАТИ ТЕСТУВАННЯ ЗАСОБУ