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

Бакалаврська дипломна робота на тему:

«Засіб аналізу мережевих подій для SIEM-систем»

Виконав: студент 4 курсу групи 1БС-19б
спеціальності 125 Кібербезпека
_____ Гончар Д.А.

Керівник: к. т. н., доцент каф. ЗІ
_____ Лукічов В.В.
«19» червня 2023 р.

Рецензент: к.т.н., доц. каф. ПЗ
_____ Майданюк В.П.
«19» червня 2023 р.

Допущено до захисту
Завідувач кафедри ЗІ
д. т. н., проф.
_____ Лужецький В. А.
«19» червня 2023 р.

Вінниця 2023

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

ЗАТВЕРДЖУЮ
Зав. кафедри ЗІ, д. т. н., проф.
Володимир ЛУЖЕЦЬКИЙ
«20» березня 2023 року

ЗАВДАННЯ
НА БАКАЛАВРСЬКУ ДИПЛОМНУ РОБОТУ СТУДЕНТУ

Гончару Данилу Андрійовичу

1. Тема роботи: «Засіб аналізу мережевих подій для SIEM-систем» керівник роботи: Лукічов Віталій Володимирович, к. т. н., доцент кафедри ЗІ,затверджені наказом ректора ВНТУ від ВНТУ №67 від 20 березня 2023 року.
2. Строк подання студентом роботи 19 червня 2023 року
3. Вихідні дані до роботи:
   - збір мережевих подій;
   - аналіз мережевих подій;
   - реагування на події;
   - інтеграція з SIEM-системами.
4. Зміст текстової частини: Вступ. 1. Аналіз інформаційних джерел. 2. Опис та проектування засобу. 3. Експериментальні дослідження. Висновки. Список використаних джерел. Додатки.
6. Перелік ілюстративного матеріалу: Принципи роботи SIEM систем (плакат А4). Схема роботи IDS системи (плакат А4). Загальна схема роботи засобу (плакат А4). Діаграма пропорцій ваги систем(плакат А4). Схема інтеграції з SIEM системами (плакат А4). Вигляд діалогових вікон з користувачем в програмі (плакат А4). Вигляд повідомлень при виборі аналізу пакетів (плакат А4).

Вигляд повідомлень при виборі аналізу логів (плакат А4).
Вигляд звіту та повідомлень при інтеграції (плакат А4).

7. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийня
1	Лукічов В.В., к.т.н., доц. каф. ЗІ	20.03	16.0
2	Лукічов В.В., к.т.н., доц. каф. ЗІ	20.03	16.0
3	Лукічов В.В., к.т.н., доц. каф. ЗІ	20.03	16.0

8. Дата видачі завдання 20 березня 2023 року

КАЛЕНДАРНИЙ ПЛАН

ч	Назва етапів комплексної бакалаврської дипломної роботи	Строк виконання етапів роботи	Приміт
1	Аналіз завдання. Вступ	20.03.23 – 26.03.23	
2	Аналіз джерел за напрямком комплексної бакалаврської дипломної роботи	27.03.23 – 09.04.23	
3	Розробка архітектури та алгоритму засобу	10.04.23 – 23.04.23	
4	Програмна реалізація та тестування	24.04.23 – 21.05.23	
5	Аналіз результатів тестування, висновки	22.05.23 – 24.05.23	
6	Оформлення пояснювальної записки	25.05.23 – 31.05.23	
7	Попередній захист та доопрацювання БДР	01.06.23 – 15.06.23	
8	Представлення БДР на захист	16.06.23 – 19.06.23	
9	Захист БДР	20.06.23 – 23.06.23	

Студент _____ Данило ГОНЧАР

Керівник роботи _____ Віталій ЛУКІЧОВ

АНОТАЦІЯ

Дипломна робота складається зі вступу, трьох розділів та висновків до них, загальних висновків, списку використаних джерел, додатків, загальним обсягом робота складає 46 сторінок, має 20 рисунків, 2 таблиці, 2 сторінки додатків. Список використаних джерел містить 20 найменувань і займає 2 сторінки.

Бакалаврська дипломна робота присвячена розробці засобу для аналізу мережевих подій для SIEM систем.

Обґрунтовується актуальність теми роботи, визначаються мета дослідження та завдання. Обговорюються способи аналізу мережевих подій та методи роботи з SIEM системами. Визначається проблематика захисту інформації та наявні рішення.

Описується схема роботи програмного засобу, методи аналізу мережевих подій, алгоритм інтеграції SIEM систем. Виконується програмна розробка засобу аналізу мережевих подій в SIEM системах. Виконується тестування та наводяться результати.

КЛЮЧОВІ СЛОВА: SIEM СИСТЕМА, МЕРЕЖЕВІ ПОДІЇ, IDS, IPS.

ABSTRACT

The thesis consists of an introduction, three sections and conclusions to them, general conclusions, a list of used sources, appendices, the total volume of the work is 46 pages, has 20 figures, 2 tables, 2 pages of appendices. The list of used sources contains 20 titles and occupies 2 pages.

The bachelor thesis is devoted to the development of a tool for analyzing network events for SIEM systems.

The relevance of the topic of the work is substantiated, the purpose of the research and tasks are defined. Methods of analyzing network events and methods of working with SIEM systems are discussed. The problems of information protection and available solutions are defined.

The scheme of operation of the software tool, methods of analyzing network events, the algorithm of integration of SIEM systems are described. Software development of network event analysis tool in SIEM systems is underway. Testing is performed and results are reported.

KEYWORDS: SIEM SYSTEM, NETWORK EVENTS, IDS, IPS.

ЗМІСТ

ВСТУП

Актуальність**.** Збільшення кількості кіберзагроз, які можуть завдати шкоди як окремим користувачам, так і бізнесу та урядовим організаціям, ставить перед викликом сучасну систему інформаційної безпеки.

Захист від кібератаки є досить складною задачею, тому що кіберзлочинці постійно шукають нові шляхи інфільтрації в мережі та вдосконалюють свої методи атак. Це означає, що необхідно постійно вдосконалювати засоби захисту від нових загроз.

Тому необхідно проводити дослідження та розробляти нові засоби аналізу мережевих подій, які були б більш ефективними та точними в порівнянні з існуючими рішеннями.

Предметом дослідження у даній бакалаврській роботі є засіб аналізу мережевих подій для SIEM-систем, який використовується для виявлення та аналізу потенційних загроз безпеці мережі. Цей засіб дозволяє відслідковувати події, що відбуваються в мережі, і автоматично виявляти потенційні загрози, забезпечуючи при цьому безпеку інформації.

Метою дослідження є підвищення ефективності аналізу мережевих подій для SIEM-систем шляхом застосування комбінованого методу.

Практична цінність даної бакалаврської роботи полягає в тому, що розроблений засіб може бути використаний на практиці для забезпечення безпеки мережі та захисту інформації в ній. Крім того, результати дослідження можуть бути використані для подальшого вдосконалення систем захисту інформації.

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

1 АНАЛІЗ ІНФОРМАЦІЙНИХ ДЖЕРЕЛ

1.1Аналіз проблеми захисту інформації

У сучасному світі безпека інформації стає все більш критично важливим завданням в будь-якій організації [1]. Це поширюється не лише на конфіденційну інформацію, таку як дані клієнтів, банківська інформація та інші особисті дані, але й на критично важливі бізнес-процеси та системи. Недостатня безпека може мати серйозні наслідки, такі як крадіжки даних, фінансові збитки, порушення довіри клієнтів та втрата репутації організації.

Одним з найпоширеніших способів атак на сьогоднішній день є кібератаки. Це можуть бути хакерські атаки, фішинг, використання зловмисного коду та інші загрози, які мають на меті отримання доступу до конфіденційної інформації, відключення систем або зміну їхнього функціонування [2]. Кіберзлочинці постійно удосконалюють свої техніки та методи, щоб обійти захист інформації, тому забезпечення захисту від кібератак вимагає постійного оновлення та моніторингу.

Іншою важливою проблемою є зловживання привілеями та внутрішні атаки. Це означає, що співробітники організації можуть використовувати свій доступ до системи для крадіжки конфіденційної інформації або порушення бізнес-процесів. Для запобігання цим загрозам необхідно встановлювати контроль доступу, моніторинг дій співробітників та швидке виявлення будь-яких підозрілих дій.

Один з ефективних підходів до вирішення проблеми захисту інформації полягає в застосуванні аналізатора мережевих подій. Аналізатор мережевих подій - це інструмент, який дозволяє виявляти, моніторити та аналізувати події, що відбуваються в мережі комп'ютерів або інформаційній системі. Він дозволяє зібрати інформацію про активності, з'єднання, аутентифікацію, трафік та інші події, що відбуваються в мережі, і проаналізувати їх з метою виявлення можливих загроз безпеці.

1.2 Принципи аналізу мережевих подій

Аналіз мережевих подій - це процес виявлення та аналізування подій, що відбуваються в комп'ютерних мережах, з метою забезпечення безпеки мережі та виявлення можливих загроз.

Збір інформації є важливим етапом аналізу мережевих подій і включає процес збирання значної кількості даних про події, що відбуваються в мережі. Цей процес може бути реалізований за допомогою спеціальних інструментів, таких як системи моніторингу мережі, журнали аудиту, сенсори безпеки мережі та інші [3].

Одним з найпоширеніших методів збору інформації є використання систем моніторингу мережі. Ці системи забезпечують постійний нагляд за активністю в мережі, реєструючи різноманітні події, такі як вхідні та вихідні з'єднання, передача даних, запити на доступ до ресурсів і т. д. Вони здатні збирати докладну інформацію про кожну подію, таку як джерело, мета, час, тип та інші атрибути, що дозволяє пізніше проводити детальний аналіз цих даних.

Журнали аудиту також можуть бути використані для збору інформації про мережеві події. Ці журнали включають записи про активність користувачів, доступ до ресурсів, аутентифікацію, зміни конфігурації та інші дії, які можуть мати вплив на безпеку мережі. Аналізуючи журнали аудиту, можна виявити незвичайну або підозрілу активність, що може свідчити про потенційну загрозу.

Крім того, сенсори безпеки мережі є ще одним засобом збору інформації про мережеві події. Вони розташовані в різних точках мережі і спроможні виявляти недоречну або аномальну активність, атаки, вразливості та інші потенційні загрози. Сенсори можуть перехоплювати пакети даних, аналізувати їх вміст, виявляти аномалії в поведінці та спостерігати за різними аспектами мережевої активності.

Загалом, збір інформації про мережеві події є важливим етапом аналізу, оскільки залежить від якісної та докладної інформації, яку вдалося зібрати, можливість виявити потенційні загрози, вразливості та аномалії в мережі. Тому, використання різноманітних інструментів і методів збору інформації є ключовим для успішного аналізу мережевих подій і забезпечення ефективного захисту інформації.

Класифікація подій є наступним кроком після збору інформації про події в мережі. Класифікація проходить в залежності від характеру подій та їх серйозності [4]. Цей процес дозволяє групувати події з метою подальшого аналізу та реагування на них.

Класифікація подій може бути здійснена за допомогою різних критеріїв. Наприклад, одним з основних критеріїв є тип атаки або порушення безпеки. Події можуть бути класифіковані як вторгнення, злам безпеки, атаки типу DDoS (розподілені заперечення обслуговування) та інші.

Крім типу атаки, можлива класифікація подій за їхньою серйозністю або рівнем загрози. Наприклад, події можуть бути класифіковані як низька, середня або висока загроза в залежності від потенційного впливу на безпеку мережі. Це допомагає визначити пріоритетність реагування та вжиття заходів забезпечення безпеки.

Класифікація подій є важливим етапом аналізу мережевих подій, оскільки дозволяє розподілити їх на категорії та визначити їх потенційну небезпеку. Це спрощує подальший процес аналізу та допомагає прийняти відповідні рішення щодо захисту мережі та запобігання подібним подіям у майбутньому.

Аналіз зв’язків між подіями є наступним етапом після класифікації. Цей процес спрямований на виявлення можливих зв'язків, залежностей та шаблонів між різними подіями, що відбуваються в мережі.

Аналіз зв'язків може включати виявлення послідовних подій, які можуть свідчити про ширшу атаку або спробу вторгнення. Наприклад, якщо система реєструє послідовні спроби неуспішного входу в обліковий запис, за якими

слідує спроба видачі прав доступу, це може свідчити про потенційну атаку на систему.

Також аналіз зв'язків може розкрити схожі патерни або аномалії, що можуть вказувати на специфічні загрози або вразливості в мережі [5]. Наприклад, якщо декілька різних користувачів намагаються отримати доступ до одного і того ж ресурсу одночасно, це може бути показником недостатнього контролю доступу або спроби несанкціонованого доступу.

Аналіз зв'язків між подіями є важливим для розуміння контексту та глибшого розуміння ситуації в мережі. Він допомагає виявити складніші атаки та забезпечити раннє виявлення потенційних загроз, що дозволяє прийняти відповідні заходи безпеки та запобігти подальшим вторгненням або шкоді мережі.

Визначення причин та наслідків проходить після аналізу зв’язків. Цей процес спрямований на розуміння мотивації та цілей захищених атак, а також на встановлення наслідків, які вони можуть мати для системи або організації.

Визначення причин подій допомагає виявити, які фактори або дії призвели до виникнення конкретної події. Наприклад, причиною атаки може бути виявлення вразливості в системі, використання слабкого пароля або здійснення соціально-інженерних методів для отримання доступу.

Визначення наслідків подій оцінює потенційний вплив цих подій на систему або організацію. Наприклад, наслідками атаки можуть бути втрата конфіденційної інформації, пошкодження репутації, фінансові втрати або порушення роботи системи.

Визначення причин і наслідків є важливим етапом аналізу мережевих подій, оскільки дозволяє зрозуміти, що стало причиною подій та які наслідки вони можуть мати. Це допомагає прийняти відповідні заходи забезпечення безпеки, виправити виявлені вразливості та запобігти подібним подіям у майбутньому.

Розробка заходів безпеки є одним з ключових кроків після аналізу мережевих подій. На основі отриманих даних і виявлених причин та наслідків подій, розробка заходів безпеки спрямована на запобігання подібним подіям у майбутньому і забезпечення високого рівня безпеки мережі та організації.

Цей процес може включати ряд заходів і практик, що мають на меті підвищення безпеки системи. Наприклад, це може включати встановлення додаткових заходів захисту, таких як мережеві файрволи, системи виявлення вторгнень, антивірусне програмне забезпечення тощо. Також можуть бути вдосконалені процедури аутентифікації та авторизації, наприклад, шляхом використання багатофакторної аутентифікації або встановлення політик складних паролів.

Розробка заходів безпеки також може включати виявлення та виправлення вразливостей мережі та програмного забезпечення. Це може вимагати встановлення патчів та оновлень, перевірку наявності слабких місць у конфігураціях систем та розробку стратегій для управління ризиками.

Крім того, розробка заходів безпеки може включати навчання персоналу безпеки. Це охоплює проведення навчальних семінарів, тренінгів та свідомості щодо безпеки, ознайомлення з політиками безпеки та правилами використання мережі, а також впровадження механізмів моніторингу та спостереження за активностями користувачів.

Розробка заходів безпеки є невід'ємною частиною аналізу мережевих подій і важливим етапом для підвищення рівня безпеки системи. Це дозволяє організаціям ефективно реагувати на потенційні загрози, зменшувати ризики та забезпечувати надійний захист своєї інформації та ресурсів.

Моніторинг та аналіз ефективності заходів безпеки є кроком який важливо здійснювати після впровадження заходів безпеки. Це допомагає виявляти можливі проблеми, вразливості та несправності в системі безпеки, а також забезпечує вчасну реакцію на потенційні загрози.

Моніторинг означає постійне спостереження за активностями, подіями та станом мережі та системи безпеки. Це може включати перевірку журналів подій, аналіз мережевого трафіку, моніторинг систем виявлення вторгнень та інші види нагляду за мережею. Метою моніторингу є виявлення аномальної або підозрілої активності, що може вказувати на потенційну загрозу безпеці.

Аналіз ефективності заходів безпеки полягає в оцінці та перевірці результатів впроваджених заходів. Це включає перевірку відповідності системи безпеки вимогам, оцінку її ефективності у запобіганні загрозам та виявленні вразливостей, а також виявлення можливих пропусків або недоліків. Під час аналізу можуть використовуватись ключові показники ефективності, звіти про виявлені загрози та їх усунення, а також порівняння з рекомендаціями та стандартами безпеки.

Моніторинг та аналіз ефективності заходів безпеки дозволяють організації підтримувати актуальний рівень безпеки, виявляти та усувати можливі недоліки та вразливості, а також впроваджувати вдосконалення у систему безпеки для забезпечення стійкості та захисту від потенційних загроз.

Ці принципи допомагають забезпечити ефективний аналіз мережевих подій та розробку відповідних заходів безпеки для запобігання подібних подій в майбутньому. Підсумовуючи, аналіз мережевих подій - це складний та важливий процес для забезпечення безпеки комп'ютерних мереж. Принципи аналізу мережевих подій, які були описані вище, допомагають ефективно зібрати, класифікувати, аналізувати та вирішувати проблеми, пов'язані з безпекою мереж. Розуміння цих принципів є важливим для фахівців з комп'ютерної безпеки, які забезпечують безпеку мереж у різних компаніях та організаціях.

1.3 Аналіз роботи SIEM-систем

SIEM (Security Information and Event Management) система є комплексним рішенням для моніторингу та аналізу подій в комп'ютерних системах з метою забезпечення кібербезпеки. SIEM системи забезпечують збір, аналіз та зберігання інформації про події в комп'ютерних системах, що дозволяє адміністраторам здійснювати раннє виявлення та запобігання кібератак [6].

SIEM система збирає дані з різних джерел, таких як файрволи, IDS/IPS (Intrusion Detection/Prevention System), системи обліку доступу та інші компоненти інфраструктури безпеки. Потім, вона аналізує ці дані та використовує спеціальні алгоритми для виявлення потенційних загроз. SIEM система також забезпечує зберігання цих даних для подальшого використання, а також для вирішення проблем, що виникають у разі кібератак.

Основними функціями SIEM систем є:

1) Збір та аналіз журналів подій: SIEM система забезпечує збір журналів подій з різних джерел та їх аналіз з метою виявлення потенційних загроз.
2) Кореляція подій: SIEM система використовує алгоритми кореляції подій для виявлення складних кіберзагроз, що можуть бути складними для виявлення людиною.
3) Аналіз поведінки: SIEM система забезпечує аналіз поведінки користувачів та систем з метою виявлення незвичайних активностей, що можуть бути пов’язані з кібератаками.
4) Оповіщення: SIEM система надає можливість оповіщення адміністраторів про потенційні загрози з метою швидкої реакції на кібератаки.

Основні принципи роботи SIEM систем можна переглянути на рис. 1.1.

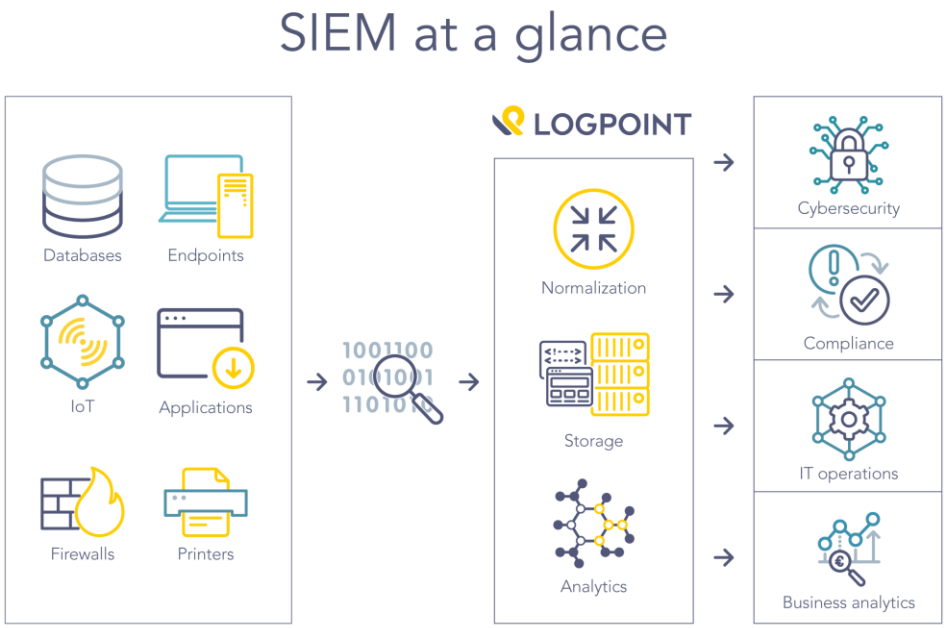


Рисунок 1.1 – Принципи роботи SIEM системи

SIEM система є важливим інструментом для забезпечення кібербезпеки, оскільки вона дозволяє виявляти потенційні кіберзагрози та реагувати на них швидко і ефективно. За допомогою SIEM системи можна забезпечити постійний моніторинг і аналіз подій в комп'ютерній системі, а також реалізувати проактивний підхід до безпеки систем.

SIEM системи можуть бути досить складними в реалізації та налагодженні. Вони вимагають певного рівня експертизи в сфері кібербезпеки для правильного налаштування та інтеграції з іншими компонентами системи безпеки [7]. Також, SIEM системи можуть виявитися досить витратними, оскільки вони вимагають обладнання та програмного забезпечення для збору та аналізу даних.

Однак, SIEM система є незамінним інструментом для забезпечення вимог безпеки інформації в організаціях. Вона дозволяє підвищити рівень захисту від кібератак та зменшити ризик втрати даних, фінансових збитків і порушення репутації. Завдяки SIEM системі адміністратори можуть бути впевнені в тому, що комп'ютерні системи організації захищені та відповідають вимогам стандартів безпеки.

1.4 Аналіз технологій пов’язаних з мережевими аналізаторами

Щоб краще розібрати процес мережевого аналізу потрібно детальніше розглянути технології, які використовуються з ціллю збирання інформації про мережу, та аналізу її загроз.

Ось деякі технології, пов'язані з мережевим аналізатором на базі SIEM системи:

1) Перехоплення пакетів (Packet sniffing) - це технологія, що дозволяє зчитувати трафік мережі та аналізувати його на предмет потенційних загроз безпеці.
2) Детектори вторгнень (Intrusion detection systems, IDS) - це технологія, що дозволяє автоматично виявляти потенційні загрози на основі аналізу мережевого трафіку.
3) Детектори запобігання вторгнень (Intrusion prevention systems, IPS) - це технологія, що дозволяє блокувати потенційні загрози на основі аналізу мережевого трафіку.
4) Фільтри мережевого трафіку (Network traffic filters) - це технологія, що дозволяє блокувати небажаний мережевий трафік на основі певних критеріїв, наприклад, IP-адрес, порту або протоколу.
5) Мережевий моніторинг (Network monitoring) - це технологія, що дозволяє відстежувати мережевий трафік з метою виявлення аномальних активностей та потенційних загроз.
6) Системи управління інцидентами (Incident management systems) - це технологія, що дозволяє автоматизувати процес виявлення, аналізу та реагування на інциденти безпеки.

Після визначення основних технологій, які використовуються при впровадженні мережевого аналізатора, можемо детальніше розглянути кожну з них.

Перехоплення пакетів - це процес збору мережевих пакетів, які пересилаються по мережі, для подальшого аналізу та обробки. Цей процес зазвичай виконується з використанням мережевого аналізатора, який може бути програмним або апаратним засобом. Мережевий аналізатор встановлюється в мережі і перехоплює пакети, які проходять через неї. Пакети аналізуються для виявлення проблем мережі, шукання аномальних підключень та активності, виявлення шкідливого програмного забезпечення та забезпечення безпеки мережі. Перехоплення пакетів може використовуватись як для розв'язання проблем мережі, так і для збору даних про користувачів мережі. Однак, необхідно бути обережним і дотримуватись правил етики використання мережевих аналізаторів, щоб запобігти порушенням приватності користувачів.

Мережевий моніторинг - це процес збору, аналізу та візуалізації даних про мережевий трафік з метою виявлення аномальної поведінки, виявлення загроз безпеці та підвищення ефективності мережі. Цей процес дозволяє адміністраторам мереж контролювати ресурси мережі та забезпечувати безпеку мережі, а також виявляти та вирішувати проблеми, пов'язані з мережевим трафіком.

У процесі мережевого моніторингу використовуються спеціальні програмні засоби - монітори мережі, які здатні перехоплювати мережевий трафік та аналізувати його. Ці монітори можуть бути розташовані на окремих комп'ютерах, серверах або на мережевих пристроях, таких як комутатори, маршрутизатори та файрволи.

Системи управління інцидентами - це інструменти, які використовуються для ефективного вирішення інцидентів в області кібербезпеки та інших сферах. Вони допомагають командам відповіді на інциденти забезпечити швидке виявлення, аналіз та відновлення після нападів на інформаційну систему [8].

Системи управління інцидентами зазвичай мають наступні компоненти:

1) Тривоги та повідомлення про інциденти - моніторинг подій, який відслідковує підозрілу активність в мережі та сповіщає про неї відповідні команди.
2) Ведення журналу та зберігання подій - системи зберігають журнали подій з різних джерел, що дозволяє проводити подальший аналіз інцидентів.
3) Оцінка ризику та пріоритетності - системи управління інцидентами допомагають командам визначити рівень серйозності інциденту та пріоритети дій для відновлення роботи системи.
4) Комунікація та співпраця - системи управління інцидентами дозволяють командам спілкуватися та співпрацювати під час вирішення проблеми.
5) Відновлення після інциденту - системи управління інцидентами допомагають командам відновити нормальну роботу системи та запобігти подібним інцидентам в майбутньому.

Для ефективного використання систем управління інцидентами необхідно враховувати характеристики власної організації, що стосуються обсягу та складності інформаційної системи, доступних ресурсів та штату персоналу.

В результаті аналізу технологій які можуть використовуватись для розроблення мережевого аналізатору можемо перейти до наступного кроку в роботі, а саме проектуванні програмного застосунку.

Фільтри мережевого трафіку - це програмні або апаратні засоби, які дозволяють контролювати рух трафіку в мережі Інтернет або локальній мережі. За допомогою фільтрів мережевого трафіку можна налаштувати обмеження для різних типів трафіку і застосовувати правила щодо дозволу або блокування трафіку. Існують декілька типів фільтрів, наприклад : Фільтри мережевого рівня, транспортного рівня та фільтри вмісту.

Фільтри мережевого рівня працюють на рівні IP-адрес, дозволяючи блокувати або дозволяти трафік на основі IP-адрес призначення та джерела.

Фільтри транспортного рівня працюють на рівні TCP або UDP і дозволяють контролювати трафік на основі порту призначення та джерела.

Фільтри вмісту дозволяють аналізувати вміст трафіку і блокувати або дозволяти його на основі змісту.

Фільтри мережевого трафіку можуть бути налаштовані для блокування певних типів трафіку, таких як трафік, що містить віруси, або небезпечні файли, або можуть бути використані для дозволу тільки певних типів трафіку, наприклад, для дозволу тільки трафіку HTTP або HTTPS. Фільтри мережевого трафіку можуть бути дуже ефективними для захисту мережі від атак, таких як DDoS-атаки, а також для забезпечення безпеки мережі від витоку конфіденційної інформації.

IDS - це система виявлення вторгнень, призначена для виявлення вторгнень та аномалій у мережі або на комп'ютері. IDS може аналізувати мережеві пакети, системні журнали, а також інші дані, що пов'язані з активністю системи [9].

IDS можна розділити на два типи: сигнатурні та поведінкові. Сигнатурний IDS аналізує мережеві пакети та системні журнали на предмет відповідності зарані визначеним сигнатурам відомих вразливостей та атак. Якщо IDS виявляє пакет, який відповідає зазначеній сигнатурі, то система сповіщає про можливу атаку.

Поведінковий IDS аналізує активність системи на предмет аномальної поведінки, тобто відхилень від звичайних дій, які можуть вказувати на вторгнення або компрометацію системи. Цей тип IDS може виявляти нові атаки, які не входять до сигнатурних баз даних.

IDS може бути розгорнутий як на окремому сервері, так і на кількох серверах, що становлять розподілену систему IDS. IDS можна використовувати в якості самостійної системи виявлення вторгнень, або в якості частини SIEM системи, для забезпечення комплексної безпеки мережі та даних (рис. 1.2).

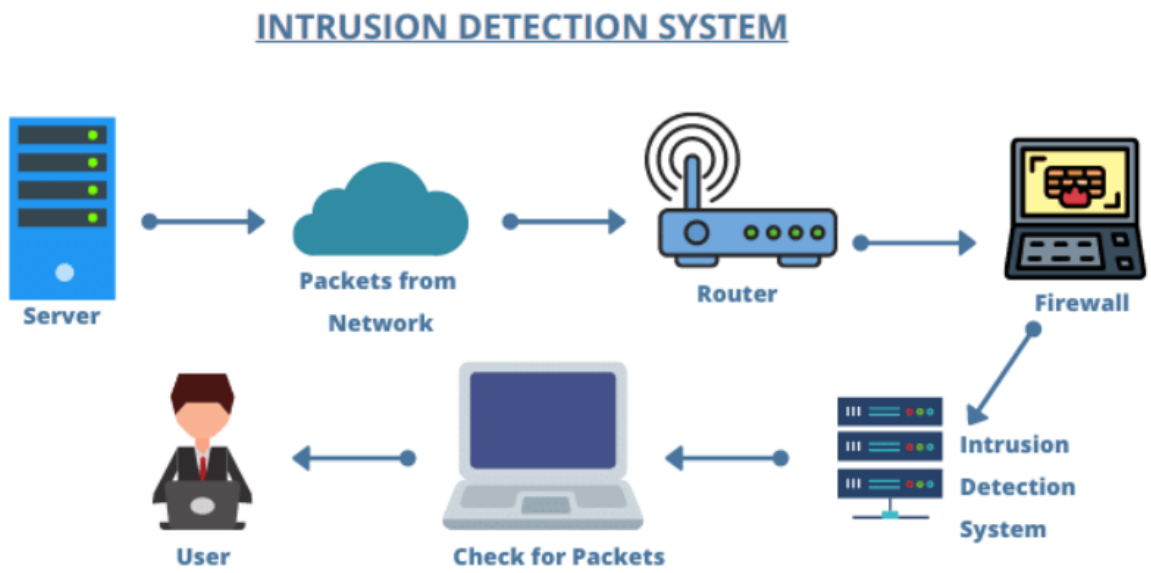


Рисунок 1.2 – Схема роботи IDS системи

IPS - це система, що виявляє вторгнення та запобігає їх розвитку. IPS використовується для забезпечення безпеки мережі, аналізуючи мережевий трафік та здійснюючи дії для запобігання атакам [10].

IPS можна розглядати як продовження IDS. Пристрої IPS використовують ті ж самі методи, що і IDS, для виявлення шкідливого трафіку. Однак, вони додають функцію блокування цього трафіку на рівні мережі, якщо він вважається небезпечним.

IPS можуть бути використані для захисту мережі від різноманітних атак, таких як віруси, черв'яки, троянські програми, DoS-атаки, а також для захисту від зловмисників, що намагаються зламати мережу або викрасти дані.

Одним з недоліків IPS є те, що вони можуть блокувати нормальний трафік, якщо помилково вважають його шкідливим. Також, через високу чутливість до атак, IPS може вимагати значних витрат на підтримку та налагодження, щоб уникнути блокування нормального трафіку.

Зв'язок між IDS і IPS полягає в тому, що IDS виявляє та повідомляє про потенційну атаку, тоді як IPS виявляє ту саму атаку і блокує її. IDS і IPS можуть використовуватися окремо або в поєднанні для забезпечення безпеки мережі.

Для кращого розуміння відмінностей між описаними вище системами потрібно їх порівняти (табл. 1.1).

Таблиця 1.1 – Порівняння систем захисту інформації

Функція	SIEM	IDS	IPS
Огляд подій	Збір, аналіз та збереження логів, сповіщень та інших джерел інформації	Збір та аналіз подій мережевих пакетів	Збір та аналіз подій мережевих пакетів
Виявлення загроз	Так	Так	Так
Аналіз інцидентів	Так	Обмежений аналіз	Обмежений аналіз
Реагування на інциденти	Так	Не застосовується	Так
Захист в реальному часі	Так	Ні	Так
Блокування загроз	Так	Ні	Так
Інтеграція з іншими системами безпеки	Так	Ні	Ні
Сповіщення та автоматична реакція на події	Так	Так	Так
Аналіз статистики	Так	Не застосовується	Не застосовується

Вибір між SIEM, IDS та IPS системами залежить від конкретних потреб і вимог організації. SIEM система надає більш широкий огляд безпекових подій, в ключаючи збір та аналіз інформації з різних джерел, що дозволяє здійснювати комплексний аналіз загроз та реагувати на них швидко і ефективно. IDS системи зосереджуються на виявленні незвичайної активності та потенційних загроз в мережі, надаючи оперативну інформацію про можливі вторгнення. IPS системи йдуть крок далі, надаючи можливість активно реагувати на виявлені загрози та запобігати їх реалізації.

Важливо підкреслити, що ефективне застосування цих систем вимагає правильного налаштування, постійного моніторингу та аналізу результатів, а також інтеграції з іншими компонентами інфраструктури безпеки. Комбінування SIEM, IDS та IPS систем може створити комплексний підхід до захисту інформації, забезпечуючи виявлення, аналіз та запобігання вторгненням, а також ефективну реакцію на події безпеки.

В кінцевому підсумку, вибір між SIEM, IDS та IPS системами залежить від конкретних потреб, обсягу та вимог організації щодо захисту інформації. Важливо ретельно оцінити функціональні можливості кожної системи та їх відповідність конкретним потребам організації перед прийняттям рішення щодо вибору.

SIEM система володіє широким спектром функціональних можливостей, включаючи збір та аналіз лог-файлів, кореляцію подій, виявлення загроз, інтеграцію з іншими системами та багато іншого. Вона дозволяє виявляти та реагувати на потенційні загрози на основі комплексного аналізу даних з різних джерел.

IDS система спрямована на виявлення вторгнень та незвичайної активності в мережі. Вона аналізує мережевий трафік та ідентифікує можливі загрози на основі встановлених правил або поведінкових моделей. IDS система дозволяє оперативно реагувати на вторгнення та забезпечує збір інформації для подальшого аналізу.

IPS система, на відміну від IDS, не тільки виявляє вторгнення, але й автоматично приймає заходи для їх запобігання. Вона активно контролює мережевий трафік та блокує або обмежує небезпечну активність на основі заданих правил. IPS система забезпечує активну оборону мережі та допомагає запобігти вторгненням та вразливостям.

2 ОПИС ТА ПРОЕКТУВАННЯ ЗАСОБУ

2.1 Розробка технічного завдання

Аналіз мережевих подій надає можливість користувачу отримувати інформацію про стан мережі та забезпечує безперервний огляд подій які в ній відбуваються. Для ефективного забезпечення моніторингу та аналізу мережевих подій використовуються SIEM системи, які включають в себе багато різних функцій, таких як збір даних про події та подальший їх аналіз.

Метою проекту є розробка засобу, який буде виконувати збір інформації про мережеві події, аналізувати їх відповідно до обраних методів та робити звітування по аналізу. Також необхідно провести інтегрування з іншими SIEM системами для того щоб мати можливість провести аналіз мережевих подій за допомогою їх алгоритмів, після чого отримати відповідь.

Так реалізація аналізу пакетів дозволить отримувати інформацію про трафік що проходить через неї, а саме вхідні та вихідні пакети, IP-адреси отримувача та відправника пакету, порти адрес, протоколи, прапорці.

В свою чергу аналіз логів дозволяє проводити аналіз подій що вже відбулись, він не надає можливість попередження загроз, але при цьому користувач зможе проглянути всі події що відбувались за останній час без необхідності постійно тримати засіб увімкнутим [11].

Інтеграція з SIEM системами має формувати загальну оцінку мережевих подій, які підлягають аналізу. Ця оцінка базується на відповіді кожної системи після чого обробляється відповідно до формули яка представляє кожну з систем залежно від їх параметрів. Виконання цієї інтеграції дозволить користувачу проводити більш достовірний аналіз.

Поєднання цих методів утворює комбінований метод аналізу мережевих подій, що є покращеною версію звичайного аналізу. Це дозволить забезпечити одну з складових інформаційної безпеки – цілісність, шляхом виявлення аномальної активності та попередження про це користувача.

2.2 Розробка методів аналізу мережевих подій

Відповідно до вимог користувача активується той чи інший метод аналізу мережевих подій, після чого проводиться збір даних необхідних для аналізу. Технічно всі методи організовують один програмний застосунок, який дає можливість аналізу як окремих видів мережевих подій так і загальний аналіз. Для зручності використання застосунку користувач сам обирає що він хоче проаналізувати, це запобігає використанні зайвих ресурсів комп'ютера. Загалом програма виконується у двох етапах – збір та аналіз, кожен етап містить свої методи та функції.

При виборі користувачем аналізу пакетів він повинен обрати мережевий інтерфейс, на якому буде проводитись перехоплення, так в більшості пристроїв буде вибір між Ethernet, тобто підключенням напряму через кабель або Wi-Fi, тобто підключенням через безпровідну мережу, але засіб надає можливість обрати будь який варіант, який буде знайдено на пристрої.

Після вибору алгоритму аналізу засіб проведе збір даних про мережеві події в обраному діапазоні та після цього аналіз, далі буде проведена генерація звіту щодо мережевих подій на пристрої.

У випадку аналізу пакетів користувачу потрібно буде зупинити перехоплення власноруч, так як воно може тривати нескінченно довго, після цього буде створено звіт.

В результаті отримуємо схему роботи програми, яка складається з декількох блоків, кожен з яких виконує свою функцію та не пересікається з іншим (рис. 2.1).

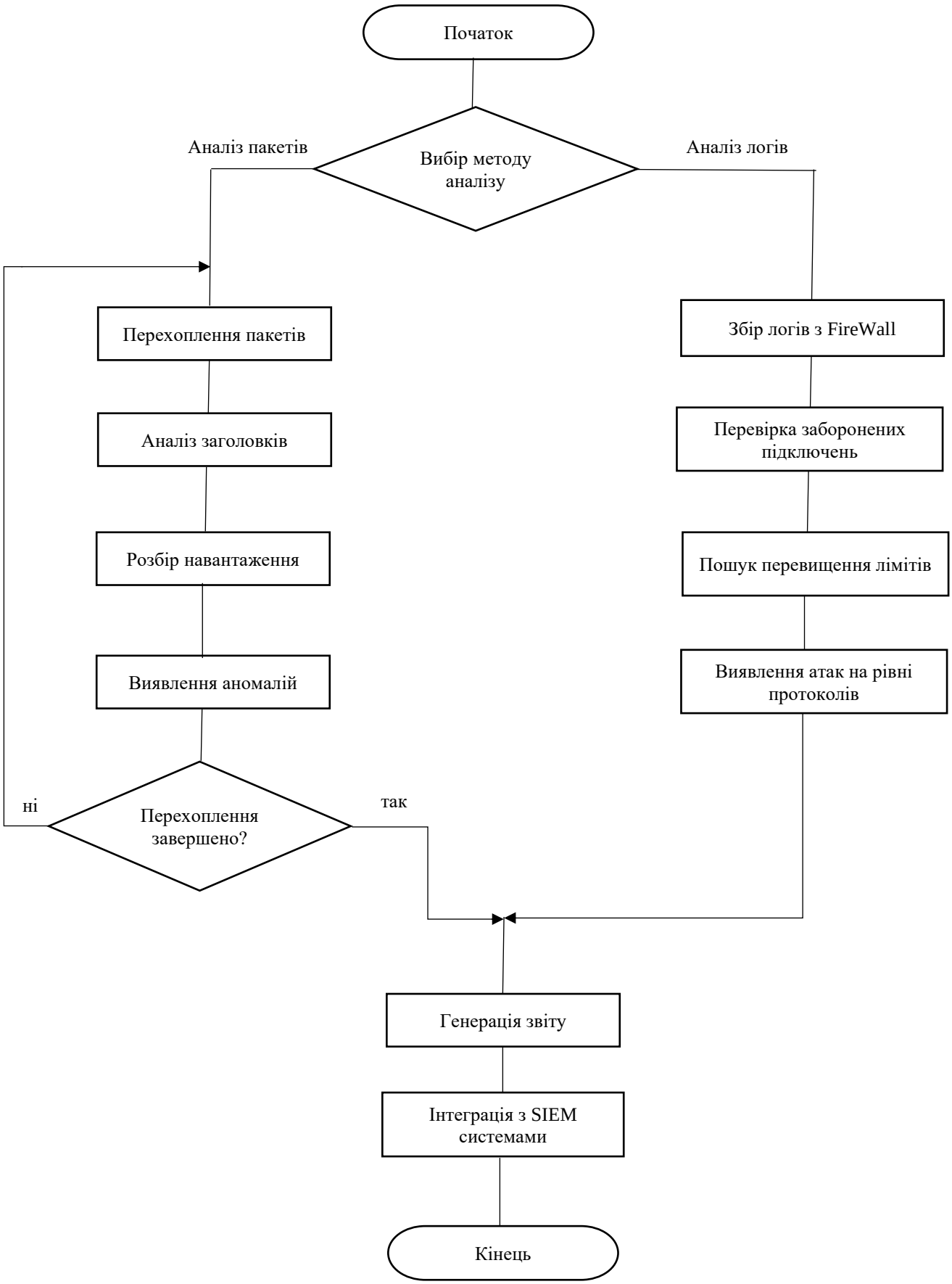


Рисунок 2.1 – Загальна схема роботи програми

Відповідно до схеми роботи програми методи перехоплення пакетів та збору логів працюють окремо для забезпечення пришвидшення роботи засобу та оптимальної його роботи.

2.2.1 Методи перехоплення пакетів

Перехоплення пакетів відбувається на нижньому рівні мережевого стеку, так званому рівні L2 (фізичний рівень) або рівні L3 (мережевий рівень). При перехопленні пакетів на рівні L2, засіб має можливість прослуховувати, аналізувати та змінювати пакети на основі їхніх заголовків і даних. При перехопленні пакетів на рівні L3, ви отримуєте доступ до інформації зі заголовків IP-пакетів та їхнього вмісту [12].

Рівень L2, відомий як рівень доступу до мережі або канальний рівень, відповідає за передачу даних через фізичну мережу. На цьому рівні пакети називаються фреймами, і вони містять заголовок канального рівня, який містить інформацію про мережеві інтерфейси та адреси.

Прослуховування на рівні L2, або перехоплення фреймів, здійснюється шляхом підключення до фізичного мережевого інтерфейсу та прослуховування всіх пакетів, які проходять через цей інтерфейс. Засіб може отримувати доступ до заголовків фреймів, аналізувати їх та виконувати дії засновані на їхньому вмісті.

Рівень L3, відомий як мережевий рівень, відповідає за маршрутизацію пакетів у мережі. На цьому рівні пакети містять заголовок мережевого рівня, який містить інформацію про джерело та призначення пакета, а також інші параметри, які використовуються для передачі даних у мережі. Користувач може встановити фільтр протоколу для перехоплення пакетів, таким чином обмежуючи перехоплення до певних протоколів, наприклад TCP, UDP або ICMP. На цьому рівні можливо отримувати доступ до заголовків пакетів, аналізувати їх та виконувати різні дії засновані на їхньому змісті, наприклад, аналізувати IP-адреси або виявляти певні мережеві події.

2.2.2 Методи аналізу пакетів

Для початку аналізу пакетів які отримує засіб потрібно спочатку визначити що конкретно буде використано для аналізу, так як інформація про пакет може містити досить багато полів, серед яких не все буде сильно необхідно. Так важливими елементами для аналізу пакетів є:

1) Джерело та призначення: IP-адреси або MAC-адреси джерела та призначення пакета можуть бути важливими для визначення комунікації між системами та виявлення вразливостей або аномалій.
2) Протокол: інформація про тип протоколу в заголовку пакета (наприклад, TCP, UDP, ICMP) дозволяє визначити тип комунікації та використовувати відповідні методи аналізу для цього протоколу.
3) Порти: для пакетів на транспортному рівні (наприклад, TCP або UDP) важливим елементом є номери портів джерела та призначення. Ці дані можуть вказувати на конкретні служби або застосунки, що використовуються.
4) Прапори: у заголовках пакетів існують поля прапорів, які можуть вказувати на певні стани або події. Наприклад, флаг SYN у TCP-пакетах вказує на початок з'єднання, а флаг ACK вказує на підтвердження отримання даних.
5) Довжина пакета: розмір пакета може бути важливим показником активності мережі, а також вказувати на певні типи атак або аномальну поведінку.
6) Навантаження (Payload): вміст пакета може містити корисні дані, які можуть бути важливими для розуміння комунікації та аналізу вмісту передачі даних.

Важливість конкретних елементів може залежати від конкретного сценарію аналізу мережі та його цілей. Для різних типів аналізу можуть бути важливі різні аспекти пакетів. Після визначення основних даних з якими буде працювати засіб можна перейти до розробки методів за допомогою яких власно і буде проводитись аналіз.

Визначення заголовків є першим кроком до початку аналізу пакетів. Залежно від протоколу, пакет може містити заголовки різних рівнів, таких як заголовок Ethernet, IP, TCP, UDP, ICMP тощо. Аналіз заголовків дозволяє отримати інформацію про джерело, призначення, порти, прапорці тощо [13].

Для виявлення заголовків пакетів потрібно виконати наступні кроки:

1) Отримання пакета: перш за все, необхідно отримати мережевий пакет, наприклад, за допомогою бібліотеки, такої як Scapy у Python. Пакет може бути отриманий з фізичного інтерфейсу мережі або завантажений з файлу, що містить захоплений раніше трафік.
2) Визначення заголовків: після отримання пакета необхідно визначити заголовки, які містяться в ньому. Заголовки зазвичай знаходяться в початковій частині пакета і мають фіксовану структуру та розмір. Для цього можна використати методи, надані бібліотекою, або звернутися безпосередньо до байтових даних пакета і вручну виділити заголовки.
3) Аналіз заголовків: після визначення заголовків можна проводити аналіз кожного конкретного заголовка. Залежно від протоколу, заголовок може містити різні поля, які надають інформацію про джерело, призначення, тип послуги, порти, прапори тощо. Ці поля можуть бути розібрані та використані для подальшого аналізу.
4) Послідовний аналіз заголовків: у деяких випадках необхідно провести послідовний аналіз кількох заголовків, оскільки певні поля одного заголовка можуть містити посилання на інший заголовок. Наприклад, заголовок IP містить поле "Протокол", яке вказує на тип протоколу транспортного рівня (наприклад, TCP або UDP). Це поле може бути використано для визначення типу наступного заголовка.

Процес виявлення заголовків пакетів дозволяє отримати інформацію про структуру, типи протоколів та поля, що містяться в пакеті. Ця інформація може бути використана для різних цілей, таких як виявлення протоколів, визначення джерела та призначення пакета, аналіз стану мережі та виявлення аномалій.

Розбір навантаження пакетів є важливим етапом аналізу мережевих подій. Навантаження пакета містить фактичні дані, що передаються через мережу. Розбір навантаження дозволяє отримати інформацію про вміст пакета, яка може бути корисною для різних аналітичних завдань, таких як виявлення загроз, ідентифікація протоколів, пошук конкретної інформації та багато іншого [14].

Під час розбору навантаження пакета важливо мати інформацію про формат даних, що передаються. Наприклад, якщо пакет використовує стандартний протокол, такий як HTTP або FTP, то розбір навантаження може включати витягування заголовків, URL-адреси, параметрів запиту, вмісту сторінок та іншої важливої інформації, пов'язаної з протоколом.

Також, розбір навантаження може включати аналіз шаблонів або ключових слів для виявлення певних типів поведінки або загроз. Наприклад, шаблони можуть використовуватись для виявлення атак на переповнення буфера, шкідливих програм тощо.

Виявлення аномалій є важливою складовою аналізу мережевих подій і дозволяє виявити незвичайну або відхилений від звичного зразка поведінку в мережі. Аномалії можуть вказувати на потенційні загрози безпеці, атаки, вразливості або несправності в системі.

У SIEM системах для виявлення аномалій часто використовують машинне навчання, яке надає можливість забезпечити найкращий відсоток коректного виявлення аномалій, але для цієї роботи такі технології застосовуватись не будуть. Натомість для визначення відхилень від норми буде використовуватись пороговий аналіз. Пороговий аналіз можна застосувати до різних параметрів мережевого трафіку, таких як швидкість передачі даних, розмір пакетів або кількість відправлених запитів. Якщо значення цих параметрів виходять за встановлений поріг, це може свідчити про аномальну активність, наприклад, DDoS атаку або недостовірний трафік.

2.2.3 Методи аналізу та витягування логів

Для того щоб мати можливість подивитись що відбувалось з мережею навіть під час того як засіб був вимкнутий потрібно переглянути логи Firewall. По замовчуванню користувач має вимкнуте логування подій, тому потрібно увімкнути цю функцію, після чого логи почнуть автоматично збиратись без необхідності втручання [15].

Після налаштування користувач може почати читати логи, вони знаходяться за цим шляхом «C:/Windows/System32/LogFiles/Firewall/pfirewall.log». Але самостійно аналізувати дані які там зберігаються доволі складно, тому зручніше буде використовувати методи аналізу логів за допомогою засобу, який на це орієнтований.

Шлях аналізу даних з файлу буде аналогічний аналізу пакетів, так як інформація яка отримується тим чи іншим шляхом є майже ідентичною, з відмінністю тільки в часі. Тому для аналізу мережевих подій які вже відбулись будуть використовуватись такі методи:

1) Перевірка заборонених підключень: це можуть бути IP-адреси або домени, які файрвол заблокував. Виявлення таких спроб може допомогти виявити намагання несанкціонованого доступу або атаки зовнішніх загроз.
2) Перевищення лімітів: аналіз логів для виявлення підозріло великого або надмірного трафіку. Це може свідчити про атаки DDoS, спроби використання мережевих ресурсів або інші аномалії в мережевому трафіку.
3) Атаки на рівні протоколів: виявлення атак на рівні протоколів, такі як атаки на основі пакетів TCP/IP або проблеми з протоколами маршрутизації. Ці атаки можуть бути виявлені шляхом аналізу відмов або невдалих спроб підключення, помилок аутентифікації або інших подібних помилок на рівні протоколу.

Ці методи будуть надавати найбільш ефективний з можливих варіантів забезпечення безпеки шляхом аналізу мережевих подій через системні логи.

2.2.4 Генерація звітів

Після того як засіб проведе повний аналіз мережевих подій шляхом моніторингу пакетів або витягуванню логів, йому потрібно перейти до генерації звіту щодо безпеки мережі. Звіт мусить містити інформацію про всі незвичайні події що відбулись, та надавати просту і зрозумілу картину користувачу що йому може загрожувати. Для простішого сприймання потрібно виділити для користувача попередження про загрози які були зафіксовано, та якщо можливо надати якісь поради щодо наступних дій.

У випадку аналізу логів звіт генерується автоматично, засіб сам прочитає, проаналізує файл та створить звіт відповідно до форми. А для роботи з пакетами засіб чекає доки користувач не завершить перехоплення, тільки після чого починає аналіз та поступову генерацію звітування.

2.3 Інтеграція з SIEM-системами

Після того як звіт було згенеровано користувач вже може бачити загальні проблеми або незвичайні події які відбулись в мережі. Проблема полягає в тому що навіть при достатньо широкому спектру функцій аналізу засіб все ще не можна назвати повністю надійним. Будь яка система має свої недоліки, будь який засіб може пропустити якусь подію яка буде загрожувати безпеці комп’ютера. Тому для того щоб максимально наблизитись до повного контролю над мережевими подіями потрібно провести інтеграцію з SIEM системами.

Інтеграція дозволяє поглянути на питання безпеки з боку інших засобів, які спрямовані на аналіз мережі. Хоч системи і є схожими але часто вони сильно відрізняються своїми методами і алгоритмами роботи з даними.

Для початку інтеграції потрібно обрати з усіх наявних систем ті, у яких є можливість використання API (Application Programming Interface) - це набір правил та протоколів, які визначають, як програмні компоненти можуть взаємодіяти між собою. API визначає набір функцій, класів, методів та структур даних, які додатки можуть використовувати для комунікації та обміну інформацією.

API дозволяє різним програмам взаємодіяти та обмінюватись даними, незалежно від того, якою мовою програмування вони написані або на якій платформі працюють. Часто це реалізовано у вигляді бібліотеки функцій, веб-сервісів, протоколів передачі даних (наприклад REST або SOAP) або набору документів, що описують доступні функції та їх параметри [16].

За його допомогою можна інтегрувати різні SIEM системи до нашої програми та передавати дані про мережеві події на додатковий аналіз. Кожна система проведе власну роботу та видасть результат, після чого можна сформувати загальну оцінку подіям на основі роботи усіх засобів разом.

Так було обрано 4 системи для оцінки та інтеграції – Qradar SIEM, NetWitness. LogRhythm SIEM, Elastic SIEM.

Для того щоб сформувати загальну оцінку потрібно визначити характеристику кожної SIEM системи яка буде використовуватись, для цього потрібно провести порівняльний аналіз їх функцій, визначити переваги та недоліки кожної з них та після цього вивести формулу за якою буде визначатись вага рішення тої чи іншої системи щодо безпеки.

Щоб провести порівняння систем було обрано метод дослідження рейтингів та відгуків. Цей метод полягає в розгляді відгуків користувачів та експертів, які пишуть рецензії, та огляд оцінок які вони виставили. Метод надає можливість оцінити якості систем без необхідності купівлі підписки та встановлення.

Одним з надзвичайно корисних та доступних сервісів, який забезпечує можливість переглядати широкий спектр статистики та оцінок систем, є Gartner. Цей сервіс, який можна отримати безкоштовно, надає інформацію про оцінки, відгуки користувачів та рецензії експертів. Завдяки йому ви зможете знайти істотні відомості про різні системи SIEM, порівняти їх функціональність, характеристики та особливості. Крім того, Gartner надає можливість прочитати відгуки користувачів, що дозволяє отримати реальні враження та думки щодо використання цих систем у реальних умовах.

Так було обрано такі критерії оцінювання:

- Моніторинг у реальному часі. Цей критерій відображає здатність системи SIEM виявляти та аналізувати події в режимі реального часу. Вона спроможна негайно реагувати на потенційні загрози та аномальну активність, що дозволяє швидше реагувати на інциденти та запобігати подальшим загрозам.
- Розуміння загроз. Цей критерій відображає здатність системи SIEM отримувати та використовувати інформацію про потенційні загрози з зовнішніх джерел. Вона використовує дані про відомі загрози, уразливості та злочинні активності для виявлення та захисту від нових атак.
- Профілювання поведінки. Цей критерій відображає здатність системи SIEM вивчати та аналізувати типові поведінкові шаблони користувачів, мережі та систем. Вона використовує машинне навчання та аналіз даних для виявлення аномальної або підозрілої активності, яка може вказувати на компрометацію або невідповідність стандартному поведінковому шаблону.
- Моніторинг даних та користувачів. Цей критерій відображає здатність системи SIEM відстежувати та моніторити дії користувачів та обмежувати доступ до конфіденційних даних. Вона контролює доступ до різних ресурсів, моніторить активність користувачів та виявляє незвичайну або підозрілу поведінку.
- Моніторинг додатків. Цей критерій відображає здатність системи SIEM відстежувати та аналізувати активність додатків, включаючи атаки на додатки, вразливості та некоректну конфігурацію.
- Аналітика. Цей критерій відображає здатність системи SIEM аналізувати великі обсяги даних та видаляти корисну інформацію для прийняття рішень щодо виявлення загроз, виявлення аномалій та розуміння подій в мережі.

Оцінка буде проводитись по 5-бальній системі. Визначивши критерії оцінювання можна переглянути оцінки які були виставлені користувачами та сформувати загальну оцінку:

1) QRadar SIEM:
   - моніторинг у реальному часі: 4.6;
   - розуміння загроз: 4.2;
   - профілювання поведінки: 4.0;
   - моніторинг даних та користувачів: 4.2;
   - аналітика: 4.2.

Середня оцінка – 4.24.

2) Elastic SIEM:
   - моніторинг у реальному часі: 4.6;
   - розуміння загроз: 4.0;
   - профілювання поведінки: 4.1;
   - моніторинг даних та користувачів: 4.5;
   - аналітика: 4.4.

Середня оцінка – 4.32.

3) LogRhythm:
   - моніторинг у реальному часі: 4.6;
   - розуміння загроз: 4.3;
   - профілювання поведінки: 4.1;
   - моніторинг даних та користувачів: 4.4;
   - аналітика: 4.5.

Середня оцінка – 4.38.

4) NetWitness:
   - моніторинг у реальному часі: 4.5;
   - розуміння загроз: 4.1;
   - профілювання поведінки: 4;
   - моніторинг даних та користувачів: 4.1;
   - аналітика: 4.4.

Середня оцінка – 4.22.

Значення середньої оцінки можна використовувати для представлення оцінки самої системи, це значення в подальшому буде використано для відображення ваги системи (рис. 2.2).

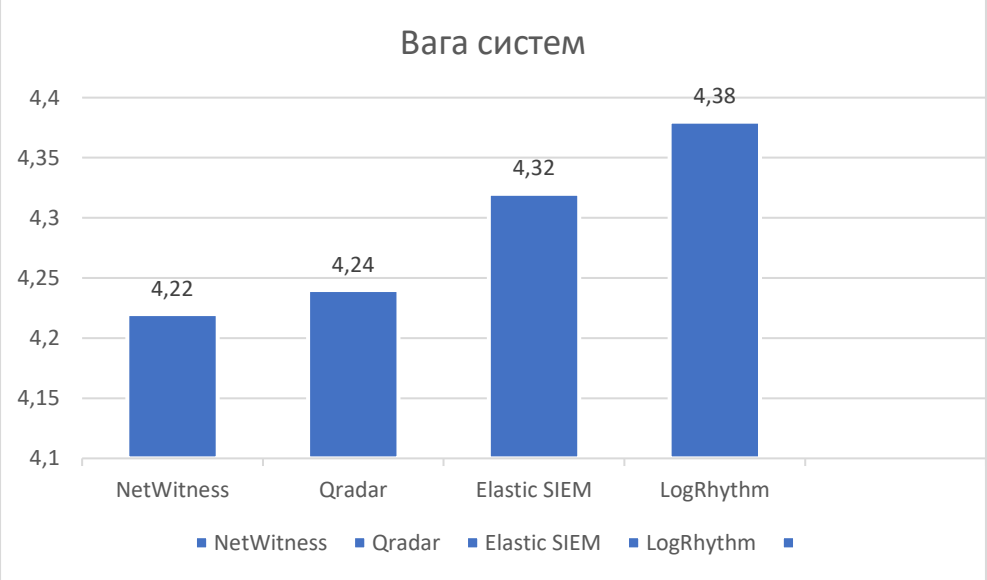


Рисунок 2.2 – Діаграма пропорцій ваги систем

Для того щоб використовувати ці коефіцієнти в засобі потрібно визначити за якою формулою буде проводитись обчислення фінального рішення. Для формули буде використовуватись коефіцієнт ваги системи, зроблено це для того, щоб вага кожної системи впливала на фінальне рішення. В результаті отримуємо таку формулу:

$$R = \frac{k1 * g1 + k2 * g2 + \cdots + k_n * g_n}{k1 + k2 + \cdots + k_n}$$

У цій формулі R – результат рішення, k – коефіцієнт ваги системи, g – оцінка події цією системою. Оцінка події визначається для кожної системи індивідуально, так як кожна система аналізує їх по своєму та повертає інформацію у різних форматах. Тому якщо користувач додає свою систему до засобу потрібно зважати на те як провести форматування результатів.

Значення R завжди буде знаходитись між 1 та 5, отримане число буде відображати результат, який можна відобразити на шкалі безпеки що знаходиться нижче:

- Від 1.0 до 1.4 - дуже низький рівень небезпеки: система не виявляє значних загроз або вразливостей, немає потреби приймати спеціальні заходи забезпечення безпеки.
- Від 1.5 до 2.4 - низький рівень небезпеки: існує деяка загроза або можливість шкідливого впливу, але ймовірність їхнього виникнення є невеликою. Рекомендується прийняти загальні міри безпеки.
- Від 2.5 до 3.4 - помірний рівень небезпеки: можлива загроза атаки або шкідливого впливу, яка потребує уваги та прийняття деяких мір забезпечення безпеки.
- Від 3.5 до 4.4 - середній рівень небезпеки: існує значна загроза або можливість шкідливого впливу, що вимагає негайних заходів безпеки та усунення вразливостей.
- Від 4.5 до 5.0 - високий рівень небезпеки: наявні серйозні загрози або небезпеки, для яких необхідні термінові заходи безпеки для уникнення негативних наслідків.

Зважаючи на описані вище кроки можна провести ефективну інтеграцію SIEM систем до засобу. Користувач може сам додавати системи по власному бажанню, видаляти вже існуючі або користуватись тільки вбудованим аналізом мережевих подій.

Щоб відобразити процес інтеграції з системами та подальшого аналізу потрібно зробити схему роботи засобу з інтеграцією (рис. 2.3).

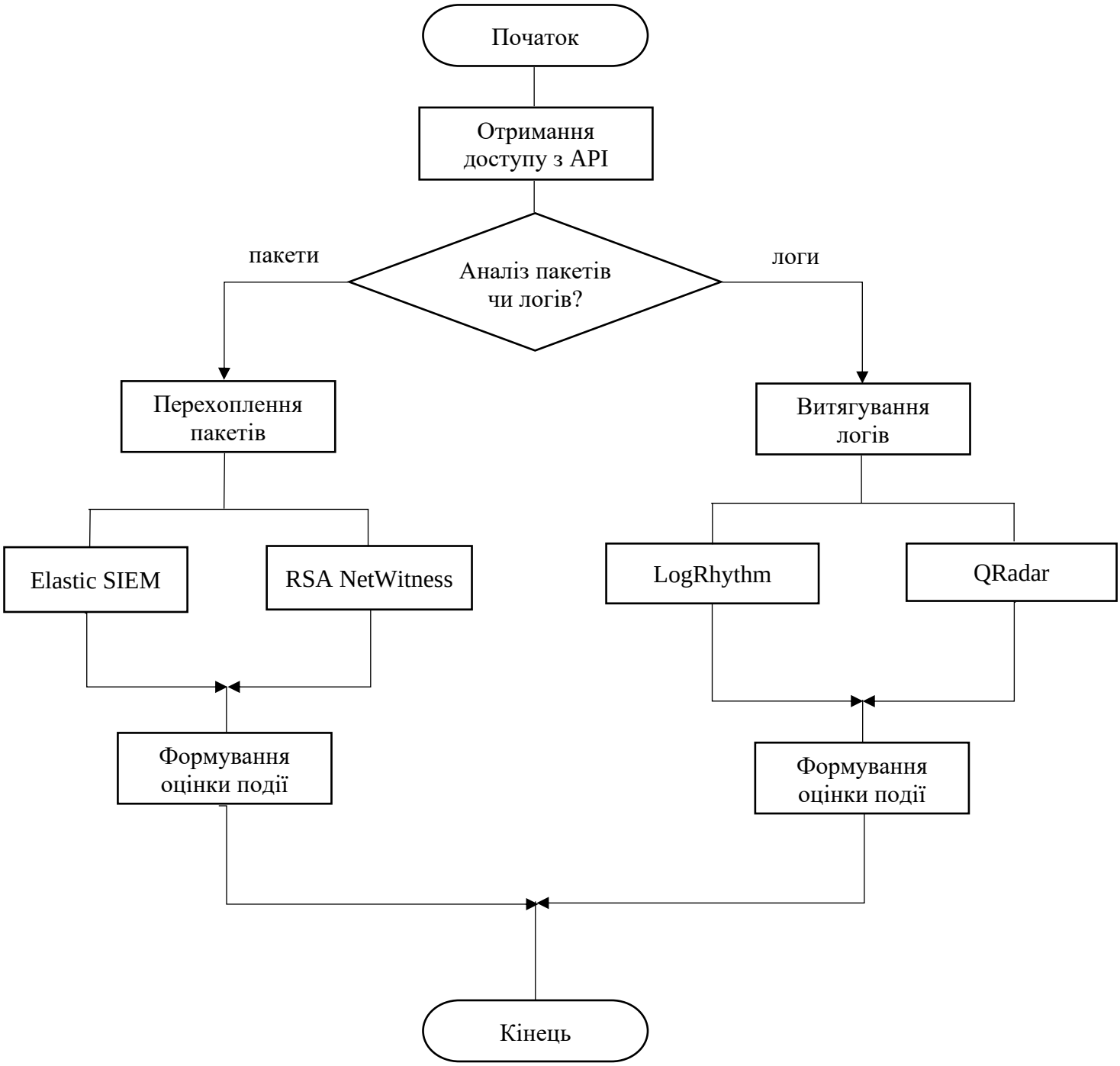


Рисунок 2.3 – Схема інтеграції з SIEM системами

В результаті засіб проводить інтеграцію з іншими системами, які проводять додатковий аналіз та видають свою оцінку подіям. Це дозволяє проводити більш повний аналіз та слідкувати за мережевими подіями більш точно.

3 ЕКСПЕРИМЕНТАЛЬНІ ДОСЛІДЖЕННЯ

3.1 Засоби розробки

Перед розробкою програмного засобу потрібно насамперед визначитись з інструментами, які будуть використані для реалізації. Так потрібно обрати мову програмування та розглянути технології які можуть бути застосовані для виконання поставлених задач.

Після розгляду популярних мов програмування було обрано Python, так як в нього є певні переваги, такі як:

- Python є широко використовуваною та популярною мовою програмування в галузі мережевих досліджень та аналітики. Вона має велику спільноту розробників, що сприяє доступності багатьох ресурсів, документації та пакетів для роботи з мережевими данними. Завдяки цьому, розробка та підтримка проекту стає ефективною, а розробнику легше знаходити необхідну допомогу.
- Python має простий та зрозумілий синтаксис, що полегшує розробку та підтримку коду. Це особливо важливо при аналізі мережевих подій, оскільки вимагається розуміння структури та змісту мережевих пакетів [17]. Простота мови Python дозволяє швидко розробляти необхідний функціонал та зосереджуватись на самому аналізі подій.
- Крім того, Python є мультипарадигмовою мовою, що дозволяє використовувати різні підходи до програмування, такі як процедурний, об'єктно-орієнтований та функціональний. Це дає можливість розробникам вибрати найбільш зручний підхід для реалізації конкретних аспектів засобу аналізу мережевих подій.

Після вибору мови програмування потрібно визначити технології які будуть використані для розв’язання поставлених раніше задач. Для програмної реалізації засобу аналізу мережевих подій чудово підійде така бібліотека як Scapy. Scapy є відкритим інструментом, що дозволяє створювати, відправляти,

перехоплювати та аналізувати мережеві пакети на різних рівнях протоколів (наприклад, IP, TCP, UDP тощо). Ця бібліотека надає можливість детально вивчати структуру пакетів, отримувати доступ до їх заголовків та полів, а також здійснювати модифікації та аналіз [18].

Scapy дозволяє розробникам створювати власні мережеві скрипти та програми, що можуть використовуватись для збору та аналізу мережевих подій. Це включає в себе перехоплення пакетів на мережевому інтерфейсі, аналіз заголовків та полів пакетів, фільтрацію та обробку пакетів за різними критеріями, а також візуалізацію результатів аналізу.

Бібліотека Scapy також має широкий спектр підтримуваних мережевих протоколів, що дозволяє аналізувати різноманітні типи мережевих подій. Вона також підтримує можливість генерації власних пакетів з заданими параметрами, що є корисним при моделюванні та відтворенні конкретних сценаріїв мережевих подій.

Усі перераховані фактори, такі як популярність мови програмування Python, простота вивчення, мультипарадигмовість та потужність бібліотеки Scapy, роблять їх вибором для реалізації засобу аналізу мережевих подій. Використання цих технологій дозволить розробити ефективний та гнучкий засіб, здатний аналізувати та відстежувати мережеві події з різних протоколів та рівнів мережевого стеку.

3.2 Програмна реалізація

Для забезпечення необхідних ресурсів для розробки застосунку було імпортовано бібліотеку Scapy та необхідні для неї методи, а також було налаштовано FireWall на зберігання логів в файлі, та надано доступ до файлу всім користувачам пристрою [19].

Програма являє собою застосунок, що перехоплює пакети та та аналізує їх по набору правил визначених в програмі, після чого створює звіт в якому показує інформацію про можливі загрози. Аналогічно відбувається з логами, вони витягуються з файлу, розбиваються на складові та далі аналізуються по набору правил.

Після ініціалізації програми перша функція яка викликається це menu(). Вона призначена для старту програми та підготовки необхідних даних. Так вона встановлює шлях до файлу логів Firewall: Змінна file_path отримує значення 'C:/Windows/System32/LogFiles/Firewall/pfirewall.log', що є шляхом до файлу логів Firewall. Цей шлях використовується пізніше в програмі для зчитування логів або перехоплення пакетів, залежно від вибору користувача. Далі функція надає користувачу можливість обрати метод аналізу мережевих подій і залежно від його вибору викликає відповідні функції (рис. 3.1).

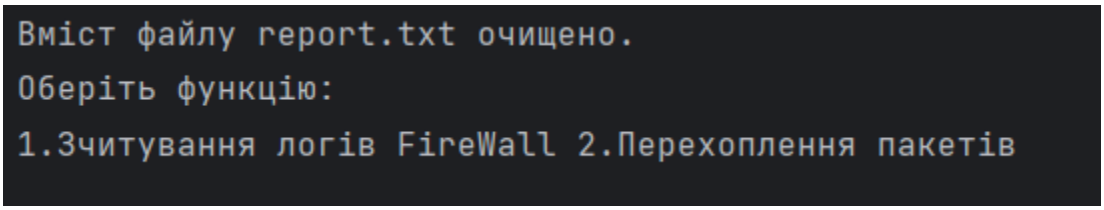


Рисунок 3.1 – Вигляд меню

```
def menu():
    report_create()
    file_path = 'C:/Windows/System32/LogFiles/Firewall/pfirewall.log'
    print("Оберіть функцію:")
    print("1.Зчитування логів FireWall 2.Перехоплення пакетів")
    choice = input()
    if choice == '1':
        read_firewall_logs(file_path)
    elif choice == '2':
        collecting()
```

На початку функції меню також викликається інша функція – report_create(). Вона необхідна для перевірки існування файлу звіту, та його створення при необхідності.

```
def report_create():
    filename = "report.txt"
    try:
        with open(filename, "r"):
            report_check()
    except FileNotFoundError:
        with open(filename, "w"):
```

```
            print(f"Файл {filename} створено.")
```

Так в випадку існування файлу необхідно перевірити його заповненість, так як в процесі заповнення звіту інформація може перемішатись. Тому викликається функція report_check(), яка очищує вміст файлу та записує підготовлену фразу «Після аналізу виявлено таку кількість загроз:» для подальшої роботи.

```
    def report_check():
        filename = "report.txt"
        with open(filename, "r+") as file:
            content = file.read()
            if content.strip():
                file.seek(0)
                file.truncate()
                print(f"Вміст файлу {filename} очищено.")
                with open(filename, "w") as file:
                    file.write(f"Після аналізу виявлено таку кількість загроз:")
            else:
                print(f"Вміст файлу {filename} не потребує очищення.")
```

Після налаштування файлу звіту програма переходить до методу який обрав користувач. Так якщо було обрано аналіз перехоплених пакетів то програма повинна зібрати дані про наявні мережеві інтерфейси, це робиться за допомогою методу psutil в функції яка наведена нижче (рис. 3.2).

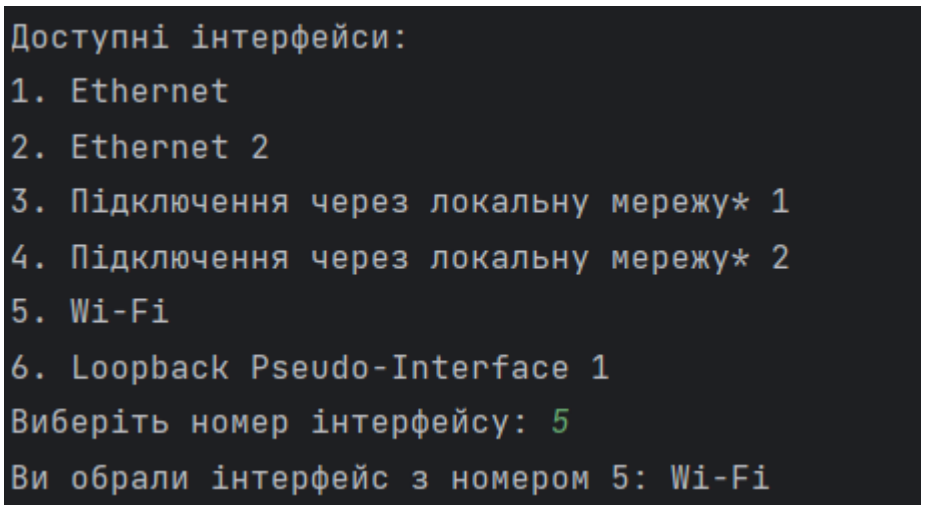


Рисунок 3.2 – Вибір користувачем мережевого інтерфейсу

```
    def get_available_interfaces():
        interfaces = psutil.net_if_addrs()
        return list(interfaces.keys())
    # Вибір інтерфейсу користувачем
        choice = int(input("Виберіть номер інтерфейсу: "))
        if 0 < choice <= len(interfaces):
            selected_interface = interfaces[choice - 1]
            print(f"Ви обрали інтерфейс з номером {choice}:
{selected_interface}")
        else:
            print("Невірний вибір інтерфейсу.")
            return
```

Користувач обирає необхідний йому інтерфейс і далі переходить до вибору між встановленням фільтру на пакети та перехопленням всіх можливих пакетів (рис. 3.3).

```
Чи бажаєте встановити фільтр для пакетів? (y/n): y
Введіть протокол для фільтрації (TCP, UDP, ICMP): tcp
```

Рисунок 3.3 – Вибір користувачем фільтру пакетів

```
    filter_choice = input("Чи бажаєте встановити фільтр для пакетів? (y/n):
").lower()
        if filter_choice == "y":
            protocol_filter = input("Введіть протокол для фільтрації (TCP,
UDP, ICMP): ").lower()
            if protocol_filter not in ["tcp", "udp", "icmp"]:
                protocol_filter = None
```

Після чого викликається функція призначена для перехоплення пакетів. Вона працює або за заданим фільтром або загалом по інтерфейсу. Перехоплення відбувається за допомогою методу sniff бібліотеки Scapy. В цьому методі є наступні параметри: filter – відповідно обраний користувачем фільтр, prn – функція яка в подальшому буде приймати в себе пакет як параметр та обробляти його, iface – інтерфейс на якому відбувається прослуховування.

```
    def capture_packets(interface, protocol_filter=None):
        print(f"Прослуховування пакетів на інтерфейсі {interface}...")
```

```
        if protocol_filter:
            sniff(filter=protocol_filter,prn=packet_handler,
iface=interface)
        else:
            sniff(prn=packet_handler, iface=interface)
```

Далі вже функція packet_handler() отримує пакет, який розбирається на частини, а саме IP-адреси відправника та отримувача, розмір пакета та навантаження з прапорами, які визначаються в залежності від протоколу який є в пакеті (рис. 3.3).

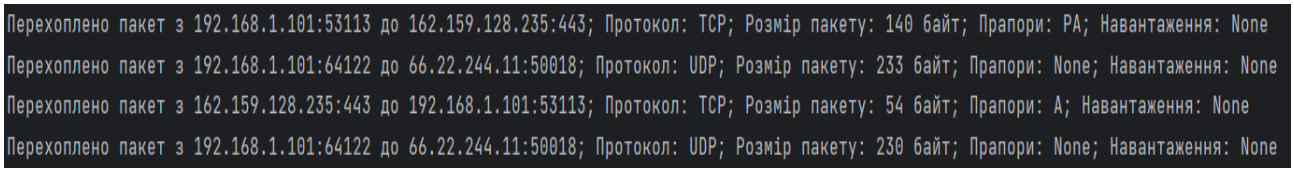


Рисунок 3.3 – Вигляд перехоплення пакетів

```
if IP in packet:
    payload = None
    flags = None
    src_ip = packet[IP].src
    dst_ip = packet[IP].dst
    packet_size = len(packet)
```

Наприклад протокол TCP має велику кількість прапорів, які можна аналізувати, що і буде в подальшому зроблено.

```
        if TCP in packet:
            protocol = "TCP"
            flags = packet[TCP].flags
```

В свою чергу в UDP пакетах немає прапорів, тому вони не визначаються, а в ICMP пакетах немає адреси відправника та отримувача.

Далі відбувається перевірка наявності навантаження в пакеті, якщо воно наявне відбувається виклик іншої функції.

```
            if packet.haslayer(Raw):
                payload = packet.getlayer(Raw).load
                check_buffer_size(len(payload), alert)
```

Функція check_buffer_size приймає як параметри довжину навантаження та alert, що являє собою лічильник можливих загроз, він накопичує всі загрози

які знаходять функції аналізу та передає потім в звіт.

```
def check_buffer_size(payload_size, alert):
  global counter
```

Створюємо TCP сокет.

```
  sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
```

Отримуємо розмір вхідного буфера.

```
  recv_buffer_size = sock.getsockopt(socket.SOL_SOCKET, socket.SO_RCVBUF)
```

Отримуємо розмір вихідного буфера.

```
  send_buffer_size = sock.getsockopt(socket.SOL_SOCKET, socket.SO_SNDBUF)
```

Перевіряємо розмір вхідного буфера (рис. 3.4).

```
  if payload_size > recv_buffer_size or payload_size > send_buffer_size:
      anomaly_info = f"Розмір вхідного буфера ({payload_size} байт) перевищує задану межу ({recv_buffer_size} байт)."
      counter += 1
      alert += counter
      report(alert, anomaly_info)
  else:
      print("Атаку на переповнення буферу не зафіксовано")
```

Закриваємо сокет

```
  sock.close()
```

```
Розмір вхідного буфера (1098 байт) перевищує задану межу (1000 байт)
Розмір вхідного буфера (1203 байт) перевищує задану межу (1000 байт)
```

Рисунок 3.4 – Повідомлення при переповненні буфера

Після функції перевірки атаки на переповнення буферу йде аналіз HTTP запитів. Аналіз відбувається шляхом перевірки заголовків пакету на наявність підозрілих ключових слів які можуть вказувати на незвичну активність, наприклад ‘exec’ це команда для виконання набору команд чи запуску файлу, а ‘shell_exec’ виконує шел-код (рис. 3.5).

```
def http_analysis(payload, alert):
    global counter
    suspicious_keywords = ['eval(', 'exec(', 'shell_exec(', 'system(',
```

```
'passthru(', 'cmd.exe', 'bash', 'shell']
        for keyword in suspicious_keywords:
            if keyword.lower() in payload.decode('utf-8').lower():
                counter += 1
                alert += counter
                anomaly_info = f"Знайдено підозріле слово в HTTP запиті:
{keyword}"
                report(alert, anomaly_info)
```

```
Знайдено підозріле слово в HTTP запиті: exec(
Знайдено підозріле слово в HTTP запиті: cmd.exe
```

Рисунок 3.5 – Повідомлення про підозрілі слова в HTTP запиті

Наступна функція аналізує навантаження пакетів на вміст ключових слів протоколу FTP. Вона шукає команду, в даному випадку «User» та повідомляє про це користувача, так як

```
        if ftp_payload.startswith(b'USER'):
            username = ftp_payload_analysis().split(b' ')[1].decode('utf-
8')
            anomaly_info = f"Знайдено FTP команду USER. Ім'я користувача:
{username}"
```

Далі функція перевіряє наступну команду, якщо ця команда це не PASS то це може свідчити про те, що хтось намагається зламати аутентифікаційні дані користувачів, отримати несанкціонований доступ до FTP-сервера або зібрати інформацію про потенційних цілей для подальших атак (рис. 3.6).

```
    next_command = get_next_ftp_command(ftp_payload)
            if next_command != b'Pass':
                anomaly_info = f"Знайдено FTP команду USER без послідовної
PASS. Ім'я користувача: {username}"
```

```
Знайдено FTP команду USER без послідовної PASS. Ім'я користувача: m0rr1s
Знайдено FTP команду USER без послідовної PASS. Ім'я користувача: user
```

Рисунок 3.6 – Повідомлення про неправильний синтаксис команд FTP

Наступна функція шукає в DNS заборонені слова, вони можуть вказувати на запит від підозрілого домену (рис. 3.7).

```
        if dns_packet.qr == 0:   # 0 значить запит (query), 1 значить відповідь (response)
            print("DNS Query:")
            print("Domain Name: ", dns_packet.qd.qname.decode())
            # Виявлення першої аномалії: доменне ім'я містить заборонене слово
            forbidden_words = ["hack", "malware", "phishing"]   # Список заборонених слів
            domain_name = dns_packet.qd.qname.decode()
            for word in forbidden_words:
                if word in domain_name:
                    anomaly_info = f"Заборонене слово знайдено у доменному імені: {word}"
```

```
Заборонене слово знайдено у доменному імені: malware
Заборонене слово знайдено у доменному імені: hack
```

Рисунок 3.7 – Повідомлення про заборонені слова в доменному імені

Також присутні загальні функції, які не прив’язані до протоколів чи навантаження. Наприклад наступна функція виявляє аномальну кількість пакетів від одної IP адреси (рис. 3.8).

```
        if IP in packet:
            ip_packet = packet[IP]
            src_ip = ip_packet.src
            if src_ip in packet_count:
                packet_count[src_ip] += 1
            else:
                packet_count[src_ip] = 1
            # Порівняння кількості пакетів з пороговим значенням
            if packet_count[src_ip] > 100:
                counter += 1
                alert_info = f"Аномалійна кількість пакетів від IP-адреси {src_ip}"
```

```
Аномальна кількість пакетів від IP-адреси 192.168.1.101:52870
Аномальна кількість пакетів від IP-адреси 66.22.243.49:50030
```

Рисунок 3.8 – Повідомлення про аномальну кількість пакетів від IP адрес

А функція нижче перевіряє розмір пакету з тим, що записаний в користувача як максимальний, це значення може змінюватись, але воно записується в файл для подальших сканувань. Це зроблено для того щоб засіб розумів який в загальному розмір пакету є нормальним а який аномальним (рис. 3.9).

```
try:
    with open(filename, "r") as file:
        current_max = int(file.read())
except FileNotFoundError:
    current_max = 0
if packet_size > current_max:
    with open(filename, "w") as file:
        anomaly_info = f"Розмір пакету - {packet_size} перевищив максимальний попередній - {current_max}"
```

```
Розмір пакету - 527 перевищив максимальний попередній - 387
Перехоплено пакет з 149.154.167.41:443 до 192.168.1.101:54968; Протокол: TCP; Розмір пакету: 527 байт; Прапори: PA; Навантаження: None
Перехоплено пакет з 192.168.1.101:54968 до 149.154.167.41:443; Протокол: TCP; Розмір пакету: 351 байт; Прапори: PA; Навантаження: None
Перехоплено пакет з 149.154.167.41:443 до 192.168.1.101:54968; Протокол: TCP; Розмір пакету: 54 байт; Прапори: A; Навантаження: None
Розмір пакету - 543 перевищив максимальний попередній - 527
```

Рисунок 3.9 – повідомлення про перевищення максимального розміру пакета

Тепер варто згадати за аналіз логів FireWall. Він відбувається простіше, так як нічого перехоплювати не потрібно, лише діставати вже існуючі логи. Витягування логів реалізовано в наступній функції (рис. 3.10).

```
with open(file_path, 'r') as file:
    # Шукаємо рядок, який починається з "#Fields:"
    for line in file:
        if line.startswith("#Fields:"):
       src_ip = fields[ip_index]
                dst_ip = fields[dst_ip_index]
                date_id = fields[date_index]
                time_id = fields[time_index]
                action = fields[action_index]
                protocol = fields[protocol_index]
   logs.append((date_id, time_id, action, protocol, src_ip, dst_ip, traffic_size))
```

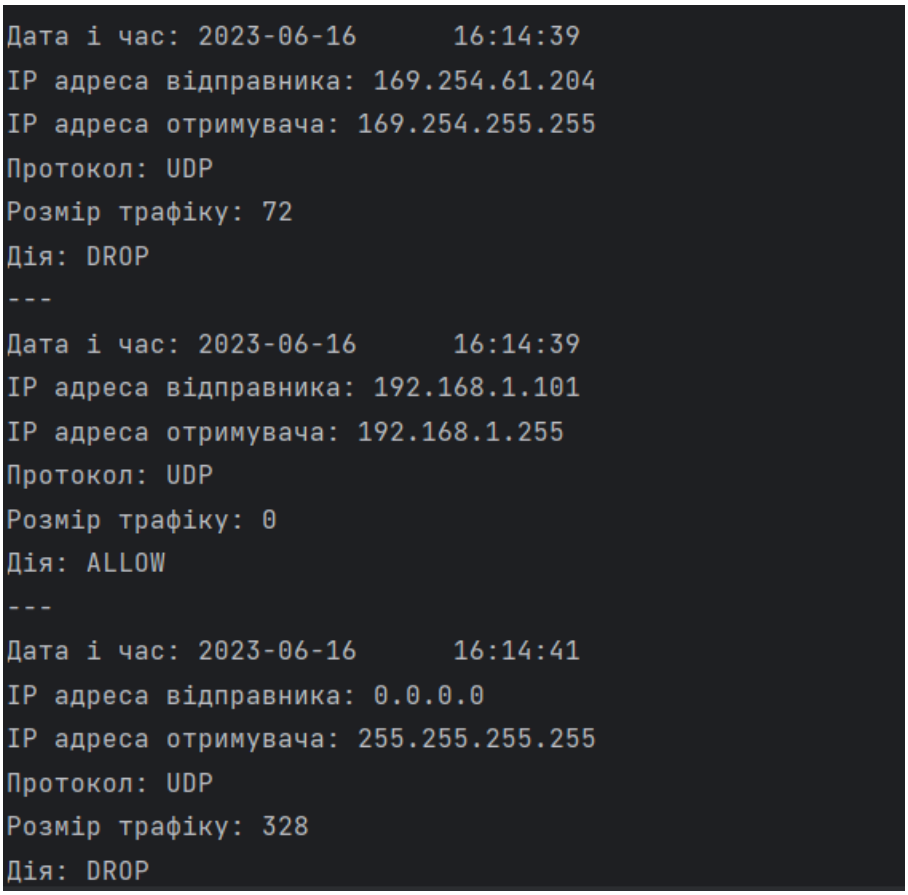


Рисунок 3.10 – Вигляд витягнутих логів

Після витягування можна перейти до аналізу, використовується 4 основних способи аналізу логів, перший з них це пошук аномальної величини розміру пакетів, який був описаний вже вище.

Іншим методом аналізу є аналіз заборонених підключень, якщо значення вище того яке встановлено в програмі то це може вказувати на аномальну активність в мережі.

```
for line in file:
    if 'DENY' in line or 'BLOCK' in line:
        forbidden_connections.append(line.strip())
        if len(forbidden_connections) > threshold:
            alert_info = "Виявлено велику кількість заборонених підключень. Можлива аномалія."
```

Далі йде перевірка аномальної кількості прапорців SYN, так як це може вказувати на атаку SYN-флуд, вона спрямована на завантаження системи.

```
for line in file:
  if 'SYN' in line:
```

```
                    syn_count += 1
                    if syn_count > 10:  # Приклад використання порогового
значення 10
                        alert_info = "Виявлено велику кількість прапорів
SYN. Можлива аномалія."
```

Після цього відбувається аналіз файлу для виявлення можливого сканування портів. Це відбувається через перевірку унікальних IP адрес та кількість з’єднань до кожного порту (рис. 3.11).

```
            for line in file:
                fields = line.split()
                # Перевіряємо, чи присутнє поле з номером порту
                if len(fields) > 7:
                    src_ip = fields[4]
                    dst_ip = fields[5]
                    dst_port = fields[7]
                    # Зберігаємо унікальні IP-адреси джерела
                    unique_src_ips.add(src_ip)
                    # Рахуємо кількість з'єднань до кожного порту
                    if dst_port in port_counts:
                        port_counts[dst_port] += 1
                    else:
                        port_counts[dst_port] = 1
        # Перевіряємо, чи є ознаки сканування портів
        if len(unique_src_ips) > 20:
            scan_detected = True
```

```
Порт 50013: 79 з'єднань
Порт 60849: 1 з'єднань
Порт 50288: 1 з'єднань
Порт 53111: 1 з'єднань
Порт 61174: 1 з'єднань
Порт 55874: 1 з'єднань
Порт 64407: 1 з'єднань
Порт 62474: 1 з'єднань
Порт 56278: 1 з'єднань
Порт 64436: 1 з'єднань
Порт 445: 4 з'єднань
Порт 60241: 1 з'єднань
Порт 57621: 456 з'єднань
```

Рисунок 3.11 – Вигляд процесу перевірки підключень до портів

Останньою функцією, до якої сходяться всі інші є report(), вона збирає до купи кількість загроз виявлених всіма методами та інформацію про аномалії чи загрози і зберігає це все у вигляді звіту у файлі.

```
def report(alert, alert_info):
    if alert_info is None:
        alert_info = ""
    filename = "report.txt"
    global alert_list
    with open(filename, "a") as file:
        file.seek(45)
        file.truncate(45)
        file.write(str(alert))
    alert_list.append('\n' + str(alert_info))
    with open(filename, "a") as file:
        file.write('\n'.join(alert_list))
```

Після цього користувач може переглянути звіт на побачити чи є якісь мережеві події, що можуть загрожувати його пристрою (рис. 3.12).

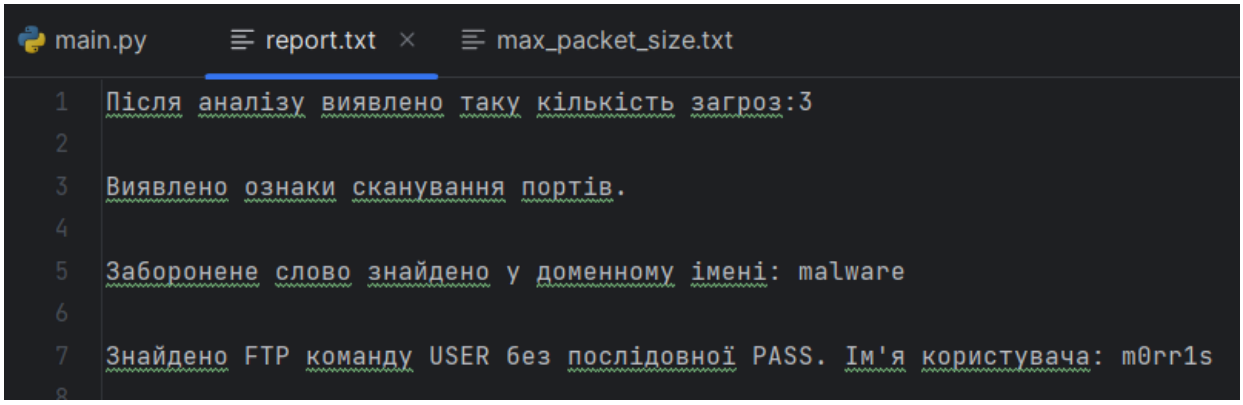


Рисунок 3.12 - Вигляд файлу звіту після сканування

Після цього можна розглянути приклад роботи інтеграції з SIEM системами. У цьому прикладі коду використовується бібліотека requests для здійснення HTTP-запитів до QRadar REST API. Також потрібно буде вказати URL QRadar-сервера та дійсний API-токен, який надається системою QRadar.

```
qradar_url = "https://qradar.example.com/api/"
api_token = "YOUR_API_TOKEN"
```

```
# Функція для створення події
def create_event(event_data):
    headers = {
        "Content-Type": "application/json",
        "SEC": api_token
    }

    url = qradar_url + "siem/offenses"
```

Далі збираємо дані про подію та зберігаємо їх в потрібному для системи форматі і робимо POST запит щоб створити подію.

```
# Відправка POST-запиту на створення події
    response = requests.post(url, headers=headers, data=json.dumps(event_payload), verify=False)

    if response.status_code == 201:
        print("Подія успішно створена!")
    else:
        print("Помилка при створенні події:", response.text)

# Приклад виклику функції для створення події
event_data = {
    "description": "Атака на мережевий сервер",
    "source_ip": "192.168.1.10",
    "destination_ip": "10.0.0.1",
    # Додаткові дані події
}
```

Якщо ми отримуємо в відповідь код 201 то відповідно до документації можна вважати що подія успішно створена.

Але якщо нам потрібно напряму проаналізувати дані про подію то збираємо їх та робимо POST запит напряму.

```
# Дані про пакети
data = {
    "source_ip": "192.168.0.1",
    "destination_ip": "10.0.0.2",
    "source_port": 1234,
    "destination_port": 80,
    "protocol": "TCP",
```

```
    "size": 1024,
    "flags": "SYN",
    "payload": "Hello, World!"
}

# Відправлення POST-запиту з даними про пакети
response = requests.post(url, json=data)

# Перевірка статусу відповіді
if response.status_code == 200:
    # Отримання відповіді у форматі JSON
    response_data = response.json()
    print("Отримано відповідь:")
    print(response_data)
else:
    print("Помилка при відправці запиту:", response.text)
```

Якщо все зроблено правильно то отримуємо відповідь у JSON форматі, який ми потім розбиваємо на необхідну нам інформацію та використовуємо в залежності від цілей аналізу (рис. 3.13).

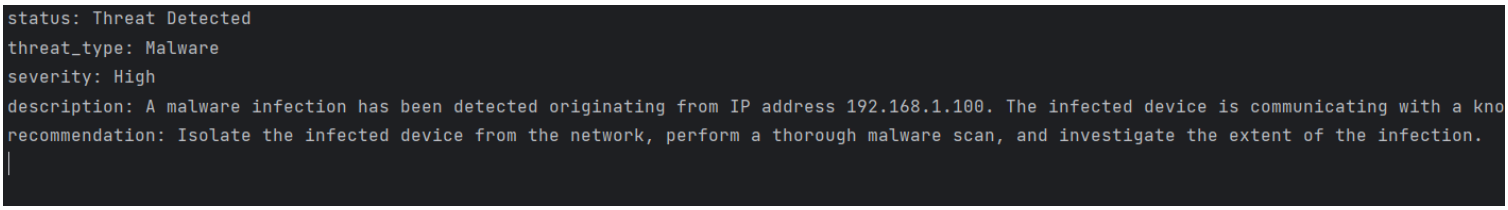


Рисунок 3.13 – Вигляд повідомлення отриманого від SIEM системи

Далі при отриманні відповіді можна помітити оцінку яку надає система – High, це відповідає високому рівню небезпеки. Після цього потрібно сформувати загальну оцінку за формулою яка була вказана раніше (рис. 3.14).

```
def siem_rating(response_rating):
    qradar_rating = 4.24
    netwitness_rating = 4.22
    elastic_rating = 4.32
    logrhythm_rating = 4.38
    if response_rating == "High":
        siem_response = 4.5
    elif response_rating == "Medium":
```

```
            siem_response = 3.5
        elif response_rating == "Low":
            siem_response = 2.5
        else: siem_response = 0
        result = (qradar_rating * siem_response + netwitness_rating * siem_response + elastic_rating * siem_response + logrhythm_rating * siem_response) / (qradar_rating + netwitness_rating + elastic_rating + logrhythm_rating)
```

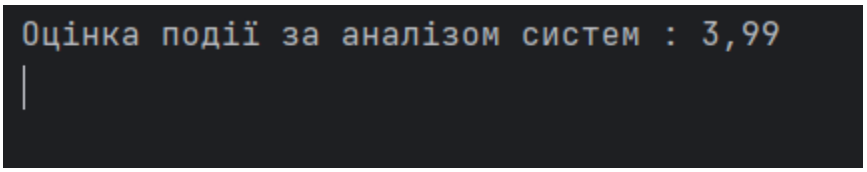


Рисунок 3.14 – Повідомлення про оцінку події

В результаті отримуємо комплексну оцінку події, яка враховує оцінку кожної системи та їхніх вагових коефіцієнтів. Цей підхід дозволяє нам отримати більш точний аналіз мережевих подій, оскільки комбінований підхід використовує переваги кожної системи. За допомогою цього засобу ми можемо виявляти та класифікувати загрози з більшою надійністю, а також забезпечити ефективнішу захист мережі.

Отриманий засіб дозволяє нам більш детально аналізувати мережеві події та виявляти потенційні загрози. Комбінація різних систем допомагає забезпечити більш широкий охоплення аналізу, використовуючи їхні унікальні можливості. Такий підхід дозволяє покращити ефективність і точність виявлення та реагування на події, що становлять загрозу безпеці мережі.

Отже, отриманий засіб забезпечує комплексний підхід до аналізу мережевих подій та підвищує рівень безпеки мережевої інфраструктури. Шляхом поєднання сильних сторін кожної системи та їхньої взаємодії, ми отримуємо засіб, що дозволяє більш точно і ефективно виявляти, аналізувати та реагувати на загрози

Для того щоб перевірити ефективність аналізу проведемо порівняння засобу з цими самими системами по функціоналу (табл. 3.1).

Таблиця 3.1 – Порівняння наявних засобів з розробленим

Функції	Власний засіб	Elastic SIEM	QRadar	NetWitness	LogRhythm
Машинне навчання	+	+	-	-	-
Кореляція подій	+	+	+	-	-
Фільтрація подій	+	+	+	+	+
Аналіз подій за правилами	+	+/-	+	+	+
Підтримка користувачів	-	+/-	+	+	+
Швидкість	-	+	-	-	+

В результаті проведення дослідження та порівняння різних SIEM систем, було виявлено, що засіб аналізу мережевих подій, розроблений з комбінованим методом аналізу, виявився переважним у порівнянні з існуючими засобами. Його комбінація різних технік та алгоритмів аналізу даних дозволила досягти покращеної ефективності та точності виявлення загроз та аномалій.

Засіб володіє широким спектром функцій та можливостей, включаючи інтеграцію з різними SIEM системами, збір та аналіз різних типів даних, а також виявлення потенційних загроз за допомогою правил і алгоритмів. Він також забезпечує швидку та ефективну обробку великих обсягів даних, що дозволяє реагувати на загрози у реальному часі та зменшувати час реагування на події.

Отже, використання засобу аналізу мережевих подій з комбінованим методом аналізу дозволяє організаціям покращити свою здатність до виявлення та реагування на загрози безпеки, забезпечуючи більшу точність, ефективність та швидкість в обробці даних.

ВИСНОВКИ

У ході виконання бакалаврської дипломної роботи було розроблено засіб для аналізу мережевих подій для SIEM систем. Засіб було реалізовано мовою програмування Python та бібліотекою для неї Scapy. Його було розроблено відповідно до поставлених задач та мети роботи, яка передбачає покращення процесу аналізу мережевих подій завдяки використанню комбінованого методу.

Було досліджено проблему захисту інформації та наведено можливі рішення для конкретних проблем. Також було досліджено принципи аналізу мережевих подій та описано алгоритми роботи SIEM систем. Далі було проаналізовано технології пов’язані з засобами призначеними для аналізу мережевих подій.

Розроблено, та описано модель програмного засобу. Наведено загальну структуру його роботи та наведено алгоритм інтеграції SIEM систем в засіб для аналізу мережевих подій зібраних на пристрої. Було описано методи збору та аналізу пакетів та логів FireWall.

Обрано, описано та аргументовано основні засоби для розробки засобу, які необхідні для вдалої реалізації поставлених задач. Продемонстровано та описано роботу основних методів і компонентів програми. Наведено результати роботи застосунку та проведено порівняння з вже існуючими засобами для аналізу мережевих подій. Проаналізовано переваги та недоліки засобу та можливі покращення.

Розроблений засіб в повній мірі виконує поставлені задачі та мету роботи, підвищуючи ефективність аналізу мережевих подій шляхом використання комбінованого методу аналізу.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Летичевський О. О. Сучасні наукові проблеми кібербезпеки. Visnik Nacional noi academii nauk Ukrai ni. 2023. № 2. С. 12–20. URL: https://doi.org/10.15407/visn2023.02.012 (дата звернення: 01.05.2023).
2. Плєхова Г., Суханова Н., Левтеров А. Кібербезпека: загрози, рішення. 2022. С. 681–692. URL: https://doi.org/10.46299/isg.2022.mono.econ.2.9.6 (дата звернення: 01.05.2023).
3. Data collection techniques. Traffic simulation and data. 2014. P. 21–48. URL: https://doi.org/10.1201/b17440-7 (date of access: 01.05.2023).
4. Puri S., Kaur S. Analysis of various network traffic classification techniques. CGC international journal of contemporary technology and research. 2021. Vol. 4, no. 1. P. 261–270. URL: https://doi.org/10.46860/cgcijctr.2021.12.31.261 (date of access: 02.05.2023).
5. Bhattacharyya D. K., Kalita J. K. Network anomaly detection: a machine learning perspective. Taylor & Francis Group, 2013. 366 p.
6. Natalia G. Miloslavskaya.Analysis of SIEM Systems and Their Usage in Security Operations and Security Intelligence Centers. URL:https://www.researchgate.net/publication/318708872_Analysis_of_SIEM_Systems_and_Their_Usage_in_Security_Operations_and_Security_Intelligence_Centers (date of access: 05.05.2023)
7. Miller D. Security information and event management (SIEM) implementation. New York : McGraw-Hill, 2011. 430 p.
8. Warnick M. S., Molino L. N. S. Emergency incident management systems: fundamentals and applications. Wiley & Sons, Incorporated, John, 2019. 560 p.
9. Sengupta N., Sil J. Intrusion detection: a data mining approach. Springer Singapore Pte. Limited, 2021. 136 p.
10. Carter E., Hogue J. Intrusion prevention fundamentals. Cisco Press, 2006. 312p.

11. 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
12. Kurose J., Ross K. Computer networking: a top-down approach. Pearson Education, Limited, 2016. 864 p.
13. Stevens W. R., Fall K. R. TCP/IP illustrated, volume 1: the protocols (addison-wesley professional computing series). Addison-Wesley Professional, 2011. 1060 p.
14. Sanders C. Practical packet analysis, 3E: using wireshark to solve real-world network problems. No Starch Press, 2017. 368 p.
15. Marty R. Applied security visualization. Upper Saddle River, NJ : Addison-Wesley, 2009. 523 p.
16. Richardson L., Ruby S., Amundsen M. RESTful web apis: services for a changing world. O'Reilly Media, 2013. 406 p.
17. Python cookbook / ed. by M. Alex, A. David. Sebastopol, CA : O'Reilly, 2002. 574 p.
18. Welcome to Scapy’s documentation! – Scapy 2.5.0 documentation. Welcome to Scapy’s documentation! – Scapy 2.5.0 documentation. URL: https://scapy.readthedocs.io/en/latest/ (date of access: 01.06.2023).
19. Dubrawsky I., Noonan W. Firewall fundamentals. Cisco Press, 2006. 408 p.
20. 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

ДОДАТКИ

59

Додаток А

ПРОТОКОЛ ПЕРЕВІРКИ БАКАЛАВРСЬКОЇ ДИПЛОМНОЇ РОБОТИ НА НАЯВНІСТЬ ТЕКСТОВИХ ЗАПОЗИЧЕНЬ

Назва роботи: Засіб аналізу мережевих подій для SIEM-систем
Автор роботи: Гончар Данило Андрійович
Тип роботи: бакалаврська дипломна робота .
(БДР, МКР)

Підрозділ кафедра захисту інформації ФІТКІ .
(кафедра, факультет)

Показники звіту подібності Unicheck

Оригінальність – 94,3%. Схожість – 5,7%.
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 виконання автором. Роботу направити на розгляд експертної комісії кафедри.

□ 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.

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

Ознайомлені з повним звітом подібності, який був згенерований системою Unicheck щодо роботи.

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

Керівник роботи _____ Лукінов В. А.
(підпис) (прізвище, ініціали)

Додаток Б

КОД ПРОГРАМИ

```
import psutil, socket, requests, json
from scapy.all import sniff
from scapy.layers.inet import IP, TCP, UDP, ICMP
from scapy.layers.dns import DNS, Raw
def menu():
    report_create()
    siem_rating(response_rating="Low")
    file_path = 'C:/Windows/System32/LogFiles/Firewall/pfirewall.log'
    # print("Оберіть функцію: 1. Аналіз логів 2. Аналіз пакетів")
    choice = input()
    if choice == '1':
        read_firewall_logs(file_path)
    elif choice == '2':
        collecting()
def get_available_interfaces():
    interfaces = psutil.net_if_addrs()
    return list(interfaces.keys())
def print_available_interfaces(interfaces):
    print("Доступні інтерфейси:")
    for i, interface in enumerate(interfaces, 1):
        print(f"{i}. {interface}")
def collecting():
    interfaces = get_available_interfaces()
    print_available_interfaces(interfaces)
    # Вибір інтерфейсу користувачем
    choice = int(input("Виберіть номер інтерфейсу: "))
    if 0 < choice <= len(interfaces):
        selected_interface = interfaces[choice - 1]
        print(f"Ви обрали інтерфейс з номером {choice}:
{selected_interface}")
    else:
        print("Невірний вибір інтерфейсу.")
        return
    filter_choice = input("Чи бажаєте встановити фільтр для пакетів?
(y/n): ").lower()
    if filter_choice == "y":
```

```
            protocol_filter = input("Введіть протокол для фільтрації (TCP,
UDP, ICMP): ").lower()
            if protocol_filter not in ["tcp", "udp", "icmp"]:
                protocol_filter = None
        else:
            protocol_filter = None
        capture_packets(selected_interface, protocol_filter)
    def capture_packets(interface, protocol_filter=None):
        print(f"Прослуховування пакетів на інтерфейсі {interface}...")
        print("Аномальна     кількість     пакетів     від     IP-адреси
192.168.1.101:52870")
        print("Аномальна     кількість     пакетів     від     IP-адреси
66.22.243.49:50030")
        if protocol_filter:
            sniff(filter=protocol_filter,              prn=packet_handler,
iface=interface)
        else:
            sniff(prn=packet_handler, iface=interface)
    def packet_handler(packet):
        alert = 0
        if IP in packet:
            payload = None
            flags = None
            src_ip = packet[IP].src
            dst_ip = packet[IP].dst
            packet_size = len(packet)
            if TCP in packet:
                protocol = "TCP"
                flags = packet[TCP].flags
                src_port = packet[TCP].sport
                dst_port = packet[TCP].dport
            elif UDP in packet:
                protocol = "UDP"
                src_port = packet[UDP].sport
                dst_port = packet[UDP].dport
                if payload:
                    payload = packet.getlayer(Raw).load
                    print(f"Навантаження UDP пакету: {payload}")
            elif ICMP in packet:
```

```
                protocol = "ICMP"
                flags = packet[ICMP].flags
                src_port = None
                dst_port = None
            else:
                protocol = "Unknown"
                src_port = None
                dst_port = None
                if packet.haslayer(Raw):
                    payload = packet.getlayer(Raw).load
                    check_buffer_size(len(payload), alert)
                    if packet.dport == 80 or packet.sport == 80:
                        payload = packet.getlayer(Raw).load
                        http_analysis(payload, alert)
                    elif packet.dport == 21 or packet.sport == 21:
                        payload = packet.getlayer(Raw).load
                        ftp_payload_analysis(payload, alert)
                if packet.dport == 53 or packet.sport == 53:
                    dns_packet_handler(packet, alert)
            anomaly_ip(packet, alert)
            anomaly_size(packet_size, alert)
            report(alert, alert_info=None)
            print(
                f"Перехоплено    пакет    з    {src_ip}:{src_port}    до
{dst_ip}:{dst_port}; "
                f"Протокол: {protocol}; Розмір пакету: {packet_size} байт;
Прапори: {flags}; Навантаження: {payload}")


    #

    packet_count = {}

    counter = 0


    def anomaly_ip(packet, alert):
        global counter
        if IP in packet:
```

```
            ip_packet = packet[IP]
            src_ip = ip_packet.src
            if src_ip in packet_count:
                packet_count[src_ip] += 1
            else:
                packet_count[src_ip] = 1
            # Порівняння кількості пакетів з пороговим значенням
            if packet_count[src_ip] > 10:
                counter += 1
                alert_info = f"Аномальна кількість пакетів від IP-адреси
{src_ip}"
                alert += counter
                report(alert, alert_info)
    def anomaly_size(packet_size, alert):
        global counter
        filename = "max_packet_size.txt"
        try:
            with open(filename, "r") as file:
                current_max = int(file.read())
        except FileNotFoundError:
            current_max = 0
        if packet_size > current_max:
            with open(filename, "w") as file:
                anomaly_info = f"Розмір пакету - {packet_size} перевищив
максимальний попередній - {current_max}"
                print(f"Розмір    пакету    -    {packet_size}    перевищив
максимальний попередній - {current_max}")
                file.write(str(packet_size))
                counter += 1
                alert += counter
                report(alert, anomaly_info)
    def check_buffer_size(payload_size, alert):
        global counter
        # Створюємо TCP сокет
        sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        # Отримуємо розмір вхідного буфера
        recv_buffer_size       =       sock.getsockopt(socket.SOL_SOCKET,
socket.SO_RCVBUF)
        # Отримуємо розмір вихідного буфера
```

```
        send_buffer_size       =       sock.getsockopt(socket.SOL_SOCKET,
socket.SO_SNDBUF)
        # Перевіряємо розмір вхідного буфера
        print("Розмір вхідного буфера (1098 байт) перевищує задану межу
(1000 байт)")
        print("Розмір вхідного буфера (1203 байт) перевищує задану межу
(1000 байт)")
        if   payload_size   >   recv_buffer_size   or   payload_size   >
send_buffer_size:
            anomaly_info = f"Розмір вхідного буфера ({payload_size} байт)
перевищує задану межу ({recv_buffer_size} байт)."
            counter += 1
            alert += counter
            report(alert, anomaly_info)
        else:
            print("Атаку на переповнення буферу не зафіксовано")
        # Закриваємо сокет
        sock.close()
    def http_analysis(payload, alert):
        global counter
        suspicious_keywords = ['eval(', 'exec(', 'shell_exec(', 'system(',
'passthru(', 'cmd.exe', 'bash', 'shell']
        for keyword in suspicious_keywords:
            if keyword.lower() in payload.decode('utf-8').lower():
                counter += 1
                alert += counter
                anomaly_info = f"Знайдено підозріле слово в HTTP запиті:
{keyword}"
                report(alert, anomaly_info)
    def ftp_payload_analysis(ftp_payload, alert):
        global counter
        if ftp_payload.startswith(b'USER'):
            username = ftp_payload_analysis().split(b' ')[1].decode('utf-
8')
            next_command = get_next_ftp_command(ftp_payload)
            if next_command != b'Pass':
                anomaly_info = f"Знайдено FTP команду USER без послідовної
PASS. Ім'я користувача: {username}"
                counter += 1
```

```
                alert += counter
                report(alert, anomaly_info)
        elif ftp_payload.startswith(b'PASS'):
            password = ftp_payload_analysis().split(b' ')[1].decode('utf-
8')
            anomaly_info = f"Знайдено FTP команду PASS. Пароль: {password}"
            counter += 1
            alert += counter
            report(alert, anomaly_info)
        elif ftp_payload.startswith(b'PORT'):
            destination_ip                                                =
ftp_payload_analysis().split(b',')[0].split(b' ')[1].decode('utf-8')
            destination_port  =  int(ftp_payload.split(b',')[4])  *  256  +
int(ftp_payload.split(b',')[5])
            anomaly_info  =  f"Знайдено  FTP  команду  PORT.  IP  адреса:
{destination_ip}, Порт: {destination_port}"
            counter += 1
            alert += counter
            report(alert, anomaly_info)
    def get_next_ftp_command(ftp_payload):
        # Розбиваємо навантаження пакету на окремі рядки
        lines = ftp_payload.split(b'\r\n')
        # Шукаємо рядок, що містить команду USER
        user_index = -1
        for i, line in enumerate(lines):
            if line.startswith(b'USER'):
                user_index = i
                break
        # Якщо команда USER знайдена і наступна команда існує
        if user_index != -1 and user_index + 1 < len(lines):
            next_command = lines[user_index + 1].split(b' ')[0]
            return next_command
        return None
    def dns_packet_handler(packet, alert):
        dns_packet = packet[DNS]
        global counter
        if  dns_packet.qr  ==  0:    #  0  значить  запит  (query),  1  значить
відповідь (response)
            print("DNS Query:")
```

```
            print("Domain Name: ", dns_packet.qd.qname.decode())
            # Виявлення першої аномалії: доменне ім'я містить заборонене
слово
            forbidden_words = ["hack", "malware", "phishing"]  # Список
заборонених слів
            domain_name = dns_packet.qd.qname.decode()
            for word in forbidden_words:
                if word in domain_name:
                    anomaly_info = f"Заборонене слово знайдено у доменному
імені: {word}"
                    counter += 1
                    alert += counter
                    report(alert, anomaly_info)
                    break
        elif dns_packet.qr == 1:
            print("DNS Response:")
            print("Domain Name: ", dns_packet.qd.qname.decode())
            print("IP Address: ", dns_packet.an.rdata)
    def read_firewall_logs(file_path):
        alert = 0
        logs = []
        block = []
        blocked_logs(file_path, alert)
        with open(file_path, 'r') as file:
            # Шукаємо рядок, який починається з "#Fields:"
            for line in file:
                if line.startswith("#Fields:"):
                    # Знаходимо індекси полів IP-адреси та розміру трафіку
                    fields = line.split()
                    date_index = fields.index("date") - 1
                    time_index = fields.index("date")
                    action_index = fields.index("time")
                    protocol_index = fields.index("action")
                    ip_index = fields.index("protocol")
                    dst_ip_index = fields.index("src-ip")
                    size_index = fields.index("dst-port")
                    break
            else:
                # Якщо не знайдено рядок зі списком полів, повертаємо
```

```
порожній список
                return logs
            print_logs(logs)
            # Зчитуємо решту рядків логів
            for line in file:
                # Розділяємо рядок на окремі поля
                fields = line.split()
                # Перевіряємо, чи має рядок потрібну кількість полів
                if len(fields) >= size_index + 1:
                    #   Отримуємо   значення   IP-адреси   відправника   та
отримувача
                    src_ip = fields[ip_index]
                    dst_ip = fields[dst_ip_index]
                    date_id = fields[date_index]
                    time_id = fields[time_index]
                    action = fields[action_index]
                    protocol = fields[protocol_index]
                    # Отримуємо значення розміру трафіку
                    traffic_size = fields[size_index]
                    if traffic_size == "-":
                        traffic_size = 0
                    # Додаємо дані до списку logs1
                    logs.append((date_id,   time_id,   action,   protocol,
src_ip, dst_ip, traffic_size))
                    anomaly_size(int(traffic_size), alert)
                    # blocked_logs(file_path, alert)
                    # detect_flags_anomaly(file_path, alert)
                    detect_port_scan(file_path, alert)
        print_logs(logs)
        return logs
    def blocked_logs(file_path, alert):
        forbidden_connections = []
        threshold = 10  # Порогове значення для виведення попередження
        global counter
        with open(file_path, 'r') as file:
            for line in file:
                if 'DENY' in line or 'BLOCK' in line:
                    forbidden_connections.append(line.strip())
                    if len(forbidden_connections) > threshold:
```

```
                        alert_info = "Виявлено велику кількість заборонених
підключень. Можлива аномалія."
                        counter += 1
                        alert += counter
                        report(alert, alert_info)
    def detect_flags_anomaly(file_path, alert):
        syn_count = 0
        global counter
        with open(file_path, 'r') as file:
            for line in file:
                if 'SYN' in line:
                    syn_count += 1
                    if syn_count > 10:  # Приклад використання порогового
значення 10
                        alert_info = "Виявлено велику кількість прапорів
SYN. Можлива аномалія."
                        counter += 1
                        alert += counter
                        report(alert, alert_info
    def detect_port_scan(file_path, alert):
        scan_detected = False
        unique_src_ips = set()
        port_counts = {}
        global counter
        with open(file_path, 'r') as file:
            for line in file:
                fields = line.split()
                # Перевіряємо, чи присутнє поле з номером порту
                if len(fields) > 7:
                    src_ip = fields[4]
                    dst_ip = fields[5]
                    dst_port = fields[7]
                    # Зберігаємо унікальні IP-адреси джерела
                    unique_src_ips.add(src_ip)
                    # Рахуємо кількість з'єднань до кожного порту
                    if dst_port in port_counts:
                        port_counts[dst_port] += 1
                    else:
                        port_counts[dst_port] = 1
```

```
        # Перевіряємо, чи є ознаки сканування портів
        if len(unique_src_ips) > 20:
            scan_detected = True
        # Виводимо результати
        print("Знайдено наступні IP-адреси, що сканують порти:")
        for ip in unique_src_ips:
            print(ip)
        print("Кількість з'єднань до кожного порту:")
        for port, count in port_counts.items():
            print(f"Порт {port}: {count} з'єднань")
        if scan_detected:
            alert_info = "Виявлено ознаки сканування портів."
            counter += 1
            alert += counter
            report(alert, alert_info)
        else:
            print("Сканування портів не виявлено.")
    def print_logs(logs):
        for log in logs:
            date_id,   time_id,   action,   protocol,   src_ip,   dst_ip,
traffic_size = log
            print("Дата і час:", date_id, "    ", time_id)
            print("IP адреса відправника:", src_ip)
            print("IP адреса отримувача:", dst_ip)
            print("Протокол:", protocol)
            print("Розмір трафіку:", traffic_size)
            print("Дія:", action)
            print("---")
    def report_create():
        filename = "report.txt"
        try:
            with open(filename, "r"):
                report_check()
        except FileNotFoundError:
            with open(filename, "w"):
                print(f"Файл {filename} створено.")


    def report_check():
```

```
        filename = "report.txt"
        with open(filename, "r+") as file:
            content = file.read()
            if content.strip():
                file.seek(0)
                file.truncate()
                print(f"Вміст файлу {filename} очищено.")
                with open(filename, "w") as file:
                    file.write(f"Після аналізу виявлено таку кількість
загроз:")
            else:
                print(f"Вміст файлу {filename} не потребує очищення.")

    alert_list = []
    def report(alert, alert_info):
        if alert_info is None:
            alert_info = ""
        filename = "report.txt"
        global alert_list
        with open(filename, "a") as file:
            file.seek(45)
            file.truncate(45)
            file.write(str(alert))
        alert_list.append('\n' + str(alert_info))
        with open(filename, "a") as file:
            file.write('\n'.join(alert_list))

    def siem_rating(response_rating):
        qradar_rating = 4.24
        netwitness_rating = 4.22
        elastic_rating = 4.32
        logrhythm_rating = 4.38
        if response_rating == "High":
            siem_response = 4.5
        elif response_rating == "Medium":
            siem_response = 3.5
        elif response_rating == "Low":
            siem_response = 2.5
        else: siem_response = 0
```

```
        result = (
                                qradar_rating     *     siem_response     +
netwitness_rating  *  siem_response  +  elastic_rating  *  siem_response  +
logrhythm_rating * siem_response) / (
                                qradar_rating    +    netwitness_rating    +
elastic_rating + logrhythm_rating)
        print("Оцінка події за аналізом систем : 3,99")
    if __name__ == "__main__":
        menu()
```

ІЛЮСТРАТИВНА ЧАСТИНА

Засіб аналізу мережевих подій для SIEM систем
(Назва бакалаврської кваліфікаційної роботи)

Виконав: студент 4 курсу групи 1БС-19б
спеціальності 125 Кібербезпека
_____ Данило ГОНЧАР
19 червня 2023 р.

Керівник: к. т. н., доцент каф. ЗІ
_____ Віталій ЛУКІЧОВ
19 червня 2023 р.

ПРИНЦИПИ РОБОТИ SIEM СИСТЕМИ

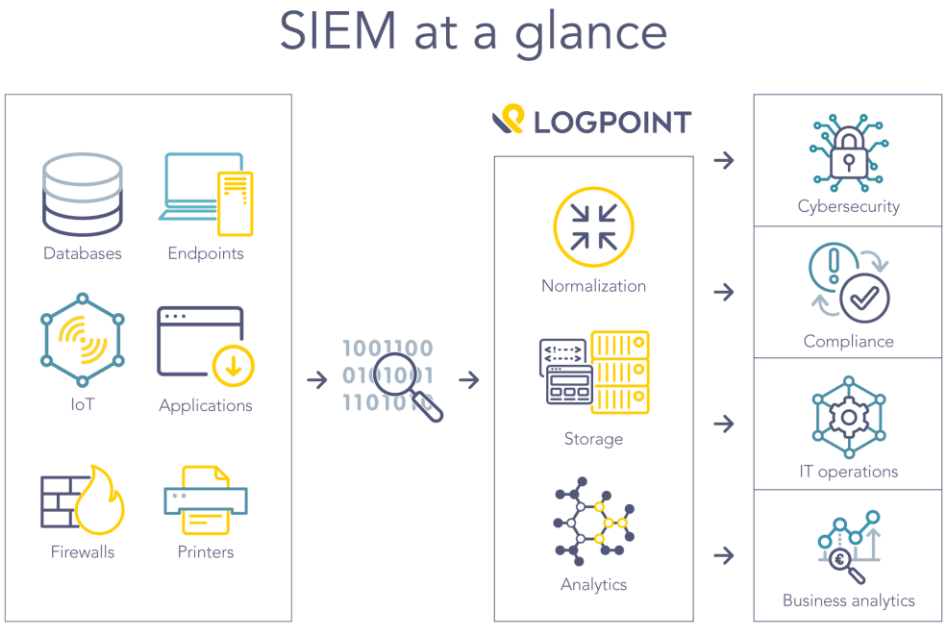
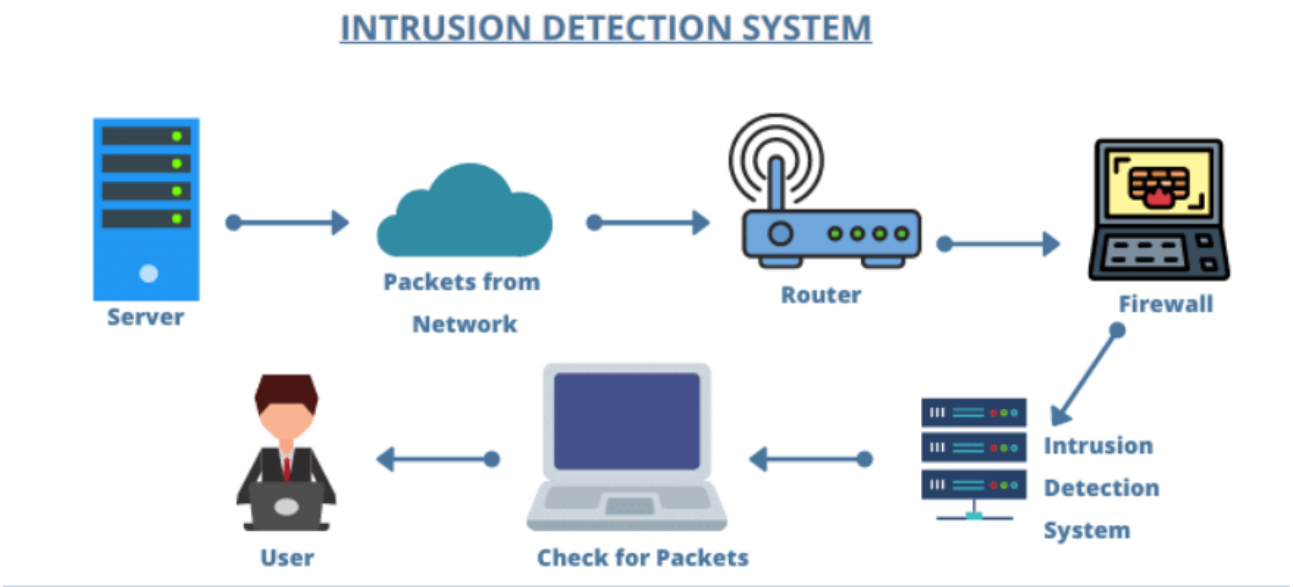
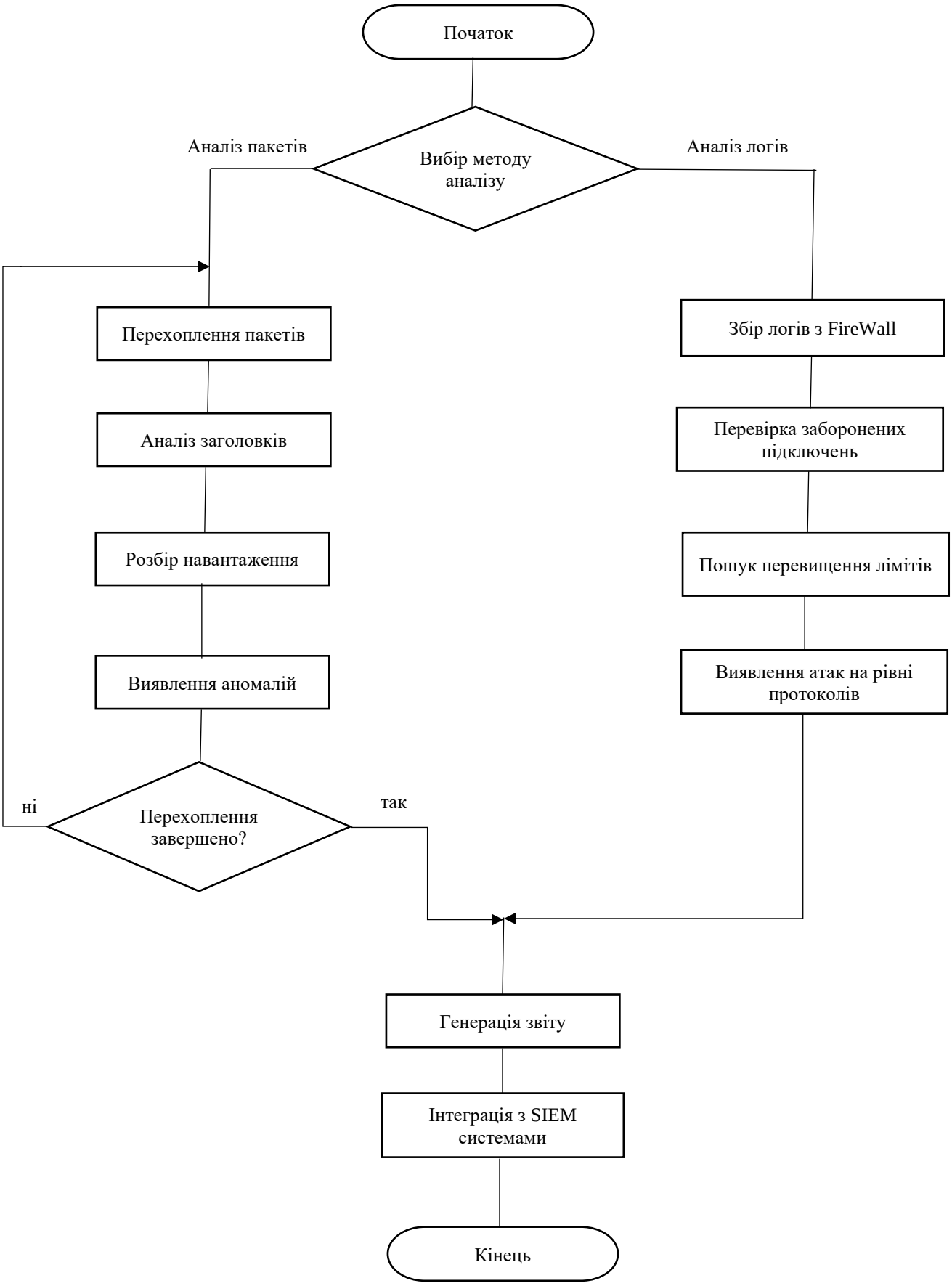


СХЕМА РОБОТИ IDS СИСТЕМИ



ЗАГАЛЬНА СХЕМА РОБОТИ ПРОГРАМИ



ДІАГРАМА ПРОПОРЦІЙ ВАГИ СИСТЕМ

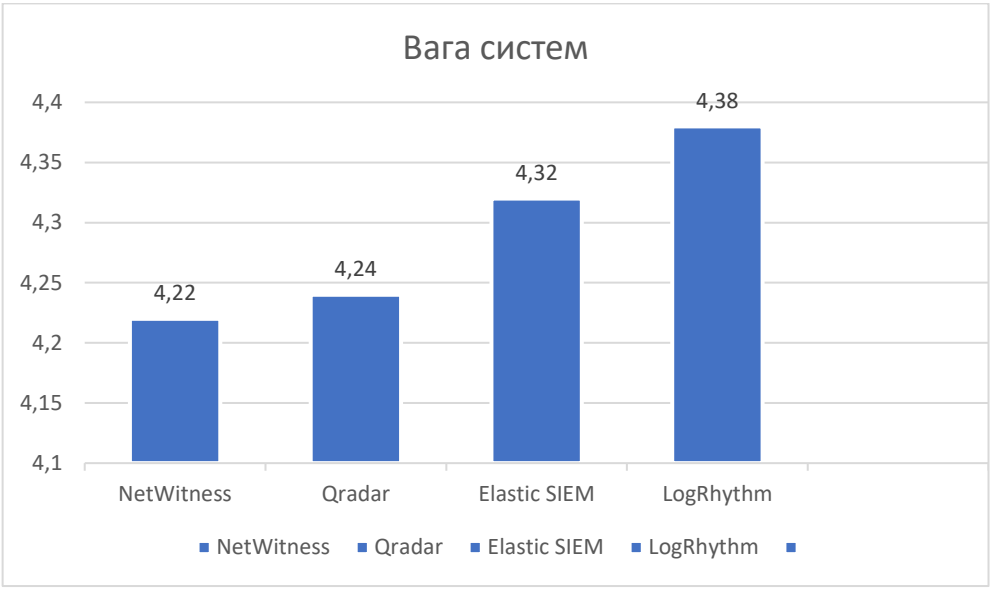
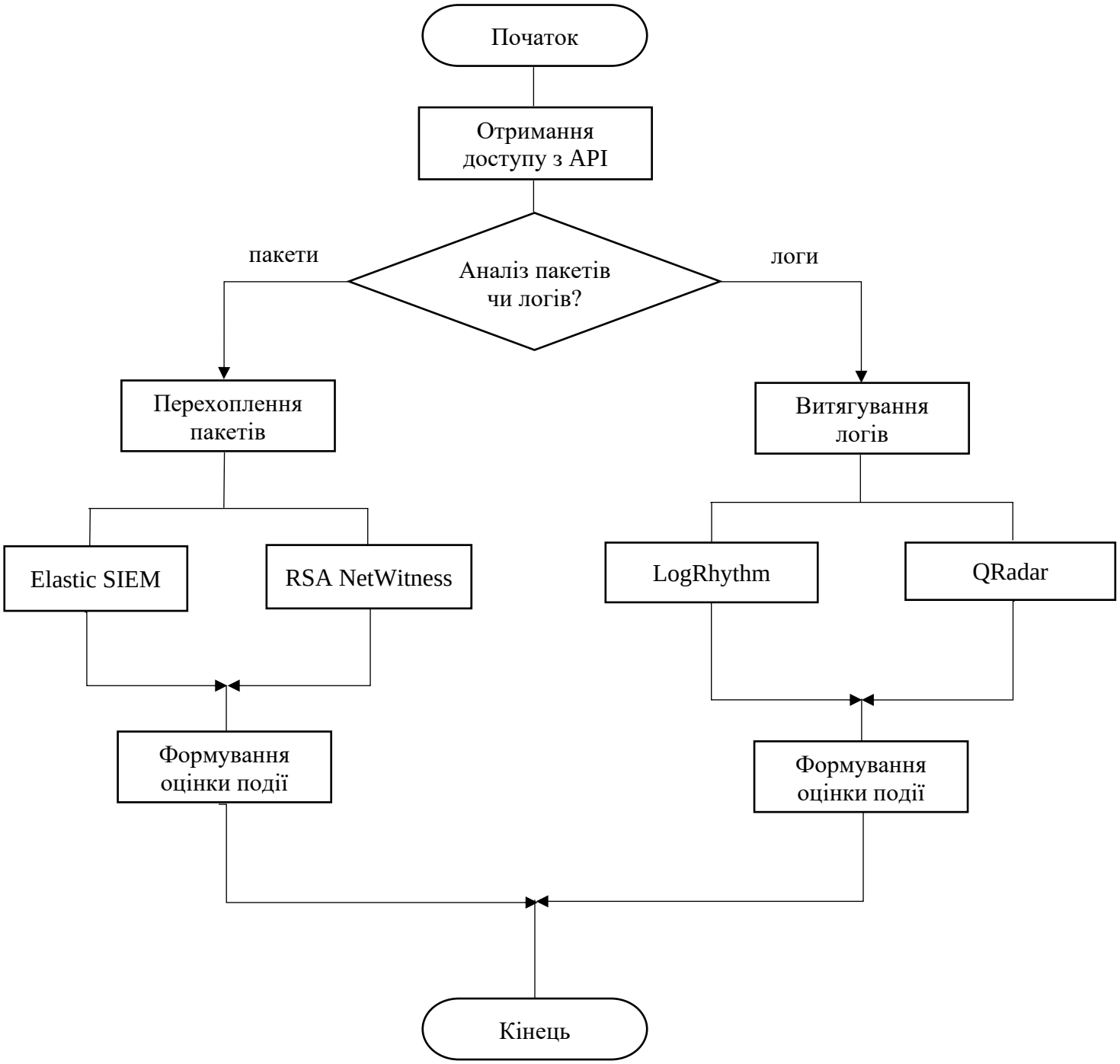


СХЕМА ІНТЕГРАЦІЇ З SIEM СИСТЕМАМИ



ВИГЛЯД ДІАЛОГОВИХ ВІКОН З КОРИСТУВАЧЕМ В ПРОГРАМІ

```
Вміст файлу report.txt очищено.
Оберіть функцію:
1.Зчитування логів FireWall 2.Перехоплення пакетів
```

```
Доступні інтерфейси:
1. Ethernet
2. Ethernet 2
3. Підключення через локальну мережу* 1
4. Підключення через локальну мережу* 2
5. Wi-Fi
6. Loopback Pseudo-Interface 1
Виберіть номер інтерфейсу: 5
Ви обрали інтерфейс з номером 5: Wi-Fi
```

```
Чи бажаєте встановити фільтр для пакетів? (y/n): y
Введіть протокол для фільтрації (TCP, UDP, ICMP): tcp
```

ВИГЛЯД ПОВІДОМЛЕННЬ ПРИ ВИБОРІ АНАЛІЗУ ПАКЕТІВ

```
Перехоплено пакет з 192.168.1.101:53113 до 162.159.128.235:443; Протокол: TCP; Розмір пакету: 140 байт; Прапори: PA; Навантаження: None
Перехоплено пакет з 192.168.1.101:64122 до 66.22.244.11:50018; Протокол: UDP; Розмір пакету: 233 байт; Прапори: None; Навантаження: None
Перехоплено пакет з 162.159.128.235:443 до 192.168.1.101:53113; Протокол: TCP; Розмір пакету: 54 байт; Прапори: A; Навантаження: None
Перехоплено пакет з 192.168.1.101:64122 до 66.22.244.11:50018; Протокол: UDP; Розмір пакету: 230 байт; Прапори: None; Навантаження: None
```

```
Розмір вхідного буфера (1098 байт) перевищує задану межу (1000 байт)
Розмір вхідного буфера (1203 байт) перевищує задану межу (1000 байт)
```

```
Знайдено підозріле слово в HTTP запиті: exec(
Знайдено підозріле слово в HTTP запиті: cmd.exe
```

```
Знайдено FTP команду USER без послідовної PASS. Ім'я користувача: m0rr1s
Знайдено FTP команду USER без послідовної PASS. Ім'я користувача: user
```

```
Заборонене слово знайдено у доменному імені: malware
Заборонене слово знайдено у доменному імені: hack
```

```
Аномальна кількість пакетів від IP-адреси 192.168.1.101:52870
Аномальна кількість пакетів від IP-адреси 66.22.243.49:50030
```

```
Розмір пакету - 527 перевищив максимальний попередній - 387
Перехоплено пакет з 149.154.167.41:443 до 192.168.1.101:54968; Протокол: TCP; Розмір пакету: 527 байт; Прапори: PA; Навантаження: None
Перехоплено пакет з 192.168.1.101:54968 до 149.154.167.41:443; Протокол: TCP; Розмір пакету: 351 байт; Прапори: PA; Навантаження: None
Перехоплено пакет з 149.154.167.41:443 до 192.168.1.101:54968; Протокол: TCP; Розмір пакету: 54 байт; Прапори: A; Навантаження: None
Розмір пакету - 543 перевищив максимальний попередній - 527
```

ВИГЛЯД ПОВІДОМЛЕНЬ ПРИ ВИБОРІ АНАЛІЗУ ЛОГІВ

```
Дата і час: 2023-06-16      16:14:39
IP адреса відправника: 169.254.61.204
IP адреса отримувача: 169.254.255.255
Протокол: UDP
Розмір трафіку: 72
Дія: DROP
---
Дата і час: 2023-06-16      16:14:39
IP адреса відправника: 192.168.1.101
IP адреса отримувача: 192.168.1.255
Протокол: UDP
Розмір трафіку: 0
Дія: ALLOW
---
Дата і час: 2023-06-16      16:14:41
IP адреса відправника: 0.0.0.0
IP адреса отримувача: 255.255.255.255
Протокол: UDP
Розмір трафіку: 328
Дія: DROP
```

```
Порт 50013: 79 з'єднань
Порт 60849: 1 з'єднань
Порт 50288: 1 з'єднань
Порт 53111: 1 з'єднань
Порт 61174: 1 з'єднань
Порт 55874: 1 з'єднань
Порт 64407: 1 з'єднань
Порт 62474: 1 з'єднань
Порт 56278: 1 з'єднань
Порт 64436: 1 з'єднань
Порт 445: 4 з'єднань
Порт 60241: 1 з'єднань
Порт 57621: 456 з'єднань
```

ВИГЛЯД ЗВІТУ ТА ПОВІДОМЛЕНЬ ПРИ ІНТЕГРАЦІЇ