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

Бакалаврська кваліфікаційна робота на тему:
«Засіб для автоматизації процесів операційного центру безпеки на основі штучного інтелекту»

Виконав: студент 4 курсу групи 1БКС–21б
спеціальності 125 Кібербезпека
_____ Олексій АНТОНЮК

Керівник: к. т. н, доцент кафедри ЗІ
_____ Леонід КУПЕРШТЕЙН
«16» червня 2025 р.

Рецензент: к. т. н. доцент каф. ПЗ
_____ Олена КОВАЛЕНКО
«16» червня 2025 р.

Допущено до захисту
Завідувач кафедри ЗІ
д. т. н., проф.
_____ Володимир ЛУЖЕЦЬКИЙ
«16» червня 2025 р.

Вінниця ВНТУ – 2025 року

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 – Кібербезпека критичних систем

ЗАТВЕРДЖУЮ
зав. кафедри ЗІ, д. т. н., проф.
Володимир ЛУЖЕЦЬКИЙ
21. березня 2025 року

ЗАВДАННЯ
НА БАКАЛАВРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

Антонюку Олексію Максимовичу

1. Тема роботи: «Засіб для автоматизації процесів операційного центру безпеки на основі штучного інтелекту»
   керівник роботи: Куперштейн Леонід Михайлович, к.т.н., доц. каф. ЗІ.
   затверджені наказом ректора ВНТУ від 20 березня 2025 року №97.
2. Строк подання студентом роботи 16 червня 2025 року.
3. Вихідні дані до роботи:
   - формат звіту - текстовий файл;
   - джерело даних - лог-файли SIEM-системи;
   - велика мовна модель – GPT-4o;
4. Зміст розрахунково-пояснювальної записки: Вступ. Аналіз предметної області. Технічне проектування застосунку. Розробка застосунку. Висновок. Перелік використаних джерел. Додатки.
5. Перелік ілюстративного матеріалу: Архітектура засобу автоматизації процесів операційного центру безпеки на основі ШІ. Схема взаємодії з джерелами даних. Алгоритм роботи програмного застосунку. Структура операційног оцентру безпеки. Вигляд згенерованого файлу. Результати тестування.

6. Консультанти розділів роботи:

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1	Куперштейн Л.М., к.т.н., доц. каф. ЗІ.	24.03.25	10.06.25
2	Куперштейн Л.М., к.т.н., доц. каф. ЗІ.	28.04.25	10.06.25
3	Куперштейн Л.М., к.т.н., доц. каф. ЗІ.	30.05.25	10.06.25

7. Дата видачі завдання 21 березня 2025 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів бакалаврської кваліфікаційної роботи	Строк виконання етапів роботи	Прим
1	Аналіз завдання. Вступ	24.03.2025 - 30.03.2025	
2	Аналіз інформаційних джерел за напрямком бакалаврської кваліфікаційної роботи	31.03.2025 - 13.04.2025	
3	Розробка рішень, моделей, алгоритмів	14.04.2025 –04.05.2025	
4	Практична реалізація, моделювання, експериментування, результати	05.05.2025 –14.05.2025	
5	Аналіз виконання БКР	15.05.2025 –18.05.2025	
6	Оформлення пояснювальної записки	19.05.2025 –26.05.2025	
7	Попередній захист БКР	27.05.2025 –30.05.2025	
8	Виправлення зауваженб, перевірка на наявність текстових запоичень, підготовка ілюстративного матеріалу	31.06.2025 –10.06.2025	
9	Представлення БКР до захисту, рецензування	11.06.2025 –13.06.2025	

Студент _____ Олексій АНТОНЮК
(підпис)

Керівник роботи _____ Леонід КУПЕРШТЕЙН
(підпис)

АНОТАЦІЯ

Бакалаврська кваліфікаційна робота складається з 60 сторінок формату А4, на яких є 13 рисунків, 2 таблиці, список використаних джерел містить 20 найменувань.

Бакалаврська робота присвячена розробці програмного засобу для автоматизації процесів операційного центру безпеки на основі штучного інтелекту.

Розроблено засіб для автоматизації процесів операційного центру безпеки на основі штучного інтелекту на мові програмування Python. Проведено тестування розробленого засобу.

Ключові слова: кібербезпека, велика мовна модель, Python, операційний центр безпеки, автоматизація.

ABSTRACT

Bachelor's qualification work consists of 60 pages of A4 format, which contains 13 drawings, 2 tables, the list of sources used contains 20 titles.

Bachelor's work is devoted to the development of software for automation of the processes of an operating center of safety based on artificial intelligence.

A tool for automation of the processes of the Operation Center of Safety based on artificial intelligence in Python programming language has been developed. Testing the developed tool was performed.

Keywords: cybersecurity, large language model, Python, security operating center, automation.

ЗМІСТ

ВСТУП

У сучасному світі обсяг кіберзагроз зростає з кожним днем, а інформаційна безпека стає критично важливою складовою будь-якої організації. Операційні центри безпеки (SOC) відіграють ключову роль у виявленні, аналізі та реагуванні на інциденти інформаційної безпеки. Проте фахівці SOC першого рівня часто стикаються з великим обсягом рутинної та повторюваної роботи, що знижує ефективність та вимагає значних людських ресурсів.

З розвитком штучного інтелекту та великих мовних моделей (LLM), з'явилася можливість частково або повністю автоматизувати частину задач першого рівня SOC. Зокрема, генерація аналітичних звітів на основі логів з SIEM-систем, фільтрація інцидентів за рівнем критичності, створення резюме подій для аналітиків та керівництва — усе це може бути делеговано інтелектуальній системі на базі LLM.

Об’єктом бакалаврської кваліфікаційної роботи є процеси операційного центру безпеки.

Предметом бакалаврської кваліфікаційної роботи є засіб автоматизації операційного центру безпеки на основі штучного інтелекту.

Метою цієї роботи є підвищення ефективності роботи операційного центру безпеки, шляхом автоматизації його процесів на основі штучного інтелекту.

Для досягнення мети було поставлено такі задачі:

- провести аналіз існуючих методів та інструментів для автоматизації процесів операційного центру безпеки
- визначити можливості штучного інтелекту в обробці та аналізу інформації з SIEM-систем про інциденти кібербезпеки
- розробити архітектуру програмного засобу
- розробити алгоритм роботи програмного засобу
- розробити програмний засіб
- провести тестування та оцінку ефективності програмного засобу

У рамках роботи було створено веб-застосунок із зручним інтерфейсом, реалізовано підключення до SIEM-системи, сформульовано оптимізовані промпти для взаємодії з мовною моделлю, а також забезпечено збереження та повторне використання звітів.

Цей програмний засіб демонструє практичну реалізацію ідеї інтеграції штучного інтелекту в повсякденну роботу SOC та може стати базою для подальших досліджень і впроваджень у сфері кібербезпеки.

1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Аналіз кіберзагроз сьогодення

За даними аналітичних видань, кількість кіберінцидентів за останні 5 років зросла вдвічі. По даним звіту QBE “Connected Business”, графік кількості кіберінцидентів зріс на 105 % у період з 2020 по 2024 рік [1]. Цифровий світ стає невід’ємною частиною життя кожного з нас. Аккаунти в соціальних мережах, банківські застосунки, нам важко представити наше життя без цих речей. Але чим більше наше життя пересікається з цифровим, тим більше ми стаємо солодкою ціллю для зловмисників, які хочуть використати вразливості та пробіли в безпеці цифрових систем для отримання несанкціонованого доступу або власної вигоди.

Однією із найбільших проблем та загроз у кіберпросторі, стали програми вимагачі. Вони отримують доступ до важливих даних на серверах організації, після чого шифрують їх та потребують викуп за повернення доступу. По звітам аналітичного звіту організації Chainalysis, в одному тільки 2023 році було зафіксовано більш ніж 3000 успішних атак з шифруванням та викупом даних. Загальна сума отриманих коштів в результаті викупів досягла понад 1.1 мільярда доларів, що стало рекордом у той час. Найбільша кількість інцидентів відбулась у сферах охорони здоров’я, осівти та енергетики. Зловмисники спеціально обирають компанії, де критична необхідність доступу до даних змушує жертв швидше погоджуватись на запропоновані умови. Якщо у 2024 році середня сума викупу була близько 760 тисяч доларів, то у 2025 вона майже перевищила відмітку у 1,2 мільйони доларів [2]. Зростання кількості атак та середньої суми викупу експерти зв’язують з поширенням так званих “as-a-service” моделей, які набирають популярність на Darknet Marketplaces. Вони дають змогу зловмисникам отримати доступ до готових інструментів та баз для атак. Одними з найуспішніших угрупувань, які спеціалізуються на програмах вимагачах, можна відмітити

LockBit, BlackCat та Conti. Для прикладу, угрупування LockBit по оцінці Malwarebytes, відповідає за більш ніж 25% усіх відомих атак з викупом та шифруванням даних за період 2023-2024 року. Окрім шифрування даних, зловмисні угрупування використовують так звану «подвійну та потрійну екстрорсію», що означає погрози що до публікації даних або атаки компаній партнерів.

Крім вимагачів, зростає кількість фішингових атак та цілеспрямованих кампаній на державну та приватну інфраструктуру. За даними звіту Verizon Data Breach Investigations Report (DBIR) 2024, середній час від моменту відкриття шкідливого листа до кліку по шкідливому посиланню або файлу, становить менш ніж 60 секунд. Зловмисники використовують генерацію персоналізованих листів з використанням штучного інтелекту та великих мовних моделей для підсилення ефекту правдоподібності повідомлень. Стандартна фішингова атака розраховує на швидку реакцію жертви в перші хвилини після отримання листа. Зловмисні угрупування часто використовують sприйоми, коли на жертву створюється емоційний тиск. Наприклад листи від генерального директора компанії, термінове оновлення ПО, або служба безпеки банку. По статистиці Proofpoint, більш ніж 84% всіх атак з отриманням несанкціонованого доступу до інфраструктури, починалися з фішингових листів, посилань та файлів [3].

Не обійшлось і без зростання кількості атак на критичну інфраструктуру. У березні 2025 року американська енергетична компанія Xcel Energy зазнала кібератаки, в результаті якої близько 65 тисяч домівок залишились без світла у штатах Техас та Нью-Мексико [4]. Причиною інциденту стали вразливості в цифрових системах контролю енергомережами. Схожі вразливості можуть бути використані в майбутньому для наступних атак. Головна привабливість критичної інфраструктури для зловмисників є критичні для життя людей та функціонування країн дані. Через що отримання доступу до даних або систем контролю дає більшу вигоду зловмисним угрупуванням. Крім того, не варто забувати про все більший перехід підприємств критичної інфраструктури на цифрові системи, це полегшує

керування та роботу персоналу та керівників підприємств. Але крім полегшення роботи та підвищення ефективності підприємств, погано захищенні системи відкривають двері як для внутрішніх зловмисників, так і для зовнішніх. Стрімкий перехід на цифрові системи керування та зберігання даних дозволяє за допомогою маленьких та незначної вразливостей отримати доступ до великої кількості даних та систем контролю, що в свою чергу полегшує роботу зловмисникам. Погана обізнаність персоналу у можливих кібератаках та витоках даних, робить їх простою мішенню для фішингових атак, або атак соціальної інженерії. За даними звіту Dragos OT Cybersecurity Year in Review, більш ніж половина інцидентів в системах управління були причиною людських помилок або недбалості. Як приклад типових помилок, можна відмітити встановлення та використання стандартних або простих паролей, встановлення та запуск не сертифікованих програмних засобів або використання робочих комп'ютерів для власних цілей.

На багатьох об'єктах критичної інфраструктури до сих пір використовуються застарілі операційні системи та програмні засоби. Для прикладу Windows XP або Windows 7, які мають відомі та не виправленні вразливості.

Крім звичних атак, з'явилась тенденція на використання штучного інтелекту та LLM моделей для швидкого виявлення вразливостей у системах, або покращення використання соціальної інженерії, наприклад допомога у написанні фішингових повідомлень або швидкий пошук даних з відкритих джерел про компанію або співробітника, які зловмисники використають для планування та виконання майбутніх атак.

Загалом, кількість та динаміка загроз у 2025 році демонструє необхідність підсилення підходів до кібербезпеки, захисту даних та доступів у кіберпросторі. Адже компаніям потрібно швидко адаптуватись до зростаючої кількості можливих атак з використанням все більш новітніших та складніших технологій [5].

Виходячи з аналізу кіберзагроз та їх зростання, компаніям потрібно сильно піклуватись про забезпечення високого рівня кіберзахисту. Крім використання сучасних фаєрволів, антивірусів, та навчання персоналу про атаки, та кіберзагрози,

потрібно мати команду активної реакції на інциденти кібербезпеки. Активна реакція на кіберінциденти забезпечить високий рівень кібезрахисту, та дозволить швидше виявити загрози та слабкі місця в захисті організації.

Команда, яка займається активним захистом та реакцію на інциденти кібезрбезпеки в межах організації називають операційним центром безпеки. Операційний центр безпеки є ключовим елементом у захисті сучасної організації від кіберзагроз. У період стрімкого розвитку цифрових технологій та зростання кількості кібератак, потреба в постійному моніторингу та аналізі подій безпеки стала критично важливою. Саме оперційний центр безпеки виконує цю функцію, забезпечуючи цілодобовий контроль за інформаційними системами організації.

Основним завданням операційного центру безпеки є виявлення, аналіз та реагування на потенційні інциденти безпеки в найкоротші терміни. Це дозволяє зменшити ймовірність серйозних наслідків від атак, таких як витік конфіденційних даних, зупинка бізнес-процесів або компрометація критичних систем. Завдяки злагодженій роботі аналітиків, операційний центр безпеки може виявити аномальну активність ще до того, як вона переросте у масштабну загрозу.

Операційний центр безпеки відіграє важливу роль і в стратегічному контексті. Він дозволяє керівництву компанії отримувати об’єктивну інформацію про стан кібербезпеки, виявляти слабкі місця у захисті, а також приймати обґрунтовані рішення щодо покращення захисних заходів. Операційний центр безпеки не лише реагує на інциденти, але й формує культуру безпеки в межах усієї організації.

Таким чином, операційний центр безпеки виступає щитом між організацією та потенційними загрозами. Він дозволяє мінімізувати ризики, забезпечити стабільність роботи інформаційних систем та зберегти довіру клієнтів і партнерів. У сучасному світі, де інформація є одним з найцінніших ресурсів, наявність ефективного операційного центру безпеки є не просто бажаною, а обов’язковою умовою для безпечного функціонування будь-якої організації.

1.2 Функції та проблематика операційного центра безпеки

Операційний центр безпеки представляє собою підрозділ компанії або цілу організаційну структуру, основна ціль якої є централізований моніторинг загроз, аналіз або реагування на інциденти інформаційної безпеки та первинна допомога або консультація співробітників компанії у разі інцидентів кібербезпеки, які пов’язанні з ними [6]. У сучасному світі операційні центри безпеки виконують ключову роль у забезпеченні високого рівня захисту кіберпростору інфраструктури. Серед основних задач операційних центрів безпеки, крім постійного моніторингу інцидентів, швидкого та своєчасного виявлення загроз також є щоденне або щотижневе звітування для контролю та підтримки високого стану захищеності інфраструктури. Через зростання кількості атак, зростає кількість вимог до операційного центру безпеки, адже саме він є центральною точкою у забезпечені кібербезпеки для компанії та її співробітників.

Спеціалісти операційного центру безпеки працюють з широким вибором інструментів, що дозволяє їм виявляти, аналізувати та реагувати на інциденти кібербезпеки. Одним із ключових засобів є SIEM-системи, які дають можливість збирати та класифікувати події з різних джерел, таких як сервери, фаєрволи, мережеве обладнання, хмарні платформи та кінцеві девайси. Найпопулярніші SIEM-системи мають назви Splunk, Wazuh, QRadar та Elastic Security. Кожна система має свої переваги та можливості. Для прикладу деякі компанії вже сьогодні добавляють модулі машинного навчання в SIEM-системи для більш чіткої класифікації подій та створення графіків.

Для роботи операційного центру безпеки, а саме реагування на часті інциденти та рутинні аналітичні дії спеціалісти використовують SOAR-платформи. SOAR рішення дозволяють автоматизувати рутинні дії, та створити сценарії реагування на інциденти по певному тригеру. Найпопулярнішими платформами станом на 2025 рік є Cortex XSOAR, Shuffle та FortiSoar. Задача архітекторів операційного центру безпеки зв’язати всі інструменти та рішення в

одну єдину систему для швидкої передачі даних з одного інструмента або віртуального носія на інший.

Операційні центри безпеки часто мають багаторівневу архітектуру. Для прикладу операційний центр першого рівня виконує першопочаткову обробку інцидентів кібербезпеки. До першого рівня відносять такі задачі як базовий аналіз, фільтрація, звітування та передача інформації на більш високі рівні. Другий рівень підтверджує інциденти, корелює їх з іншими джерелами та проводить глибоку аналітику. Третій рівень займається спеціалізованою аналітикою та зворотнім інжинирінгом.

Схема розподілення обов’язків по рівням представленна на рисунку 1.1

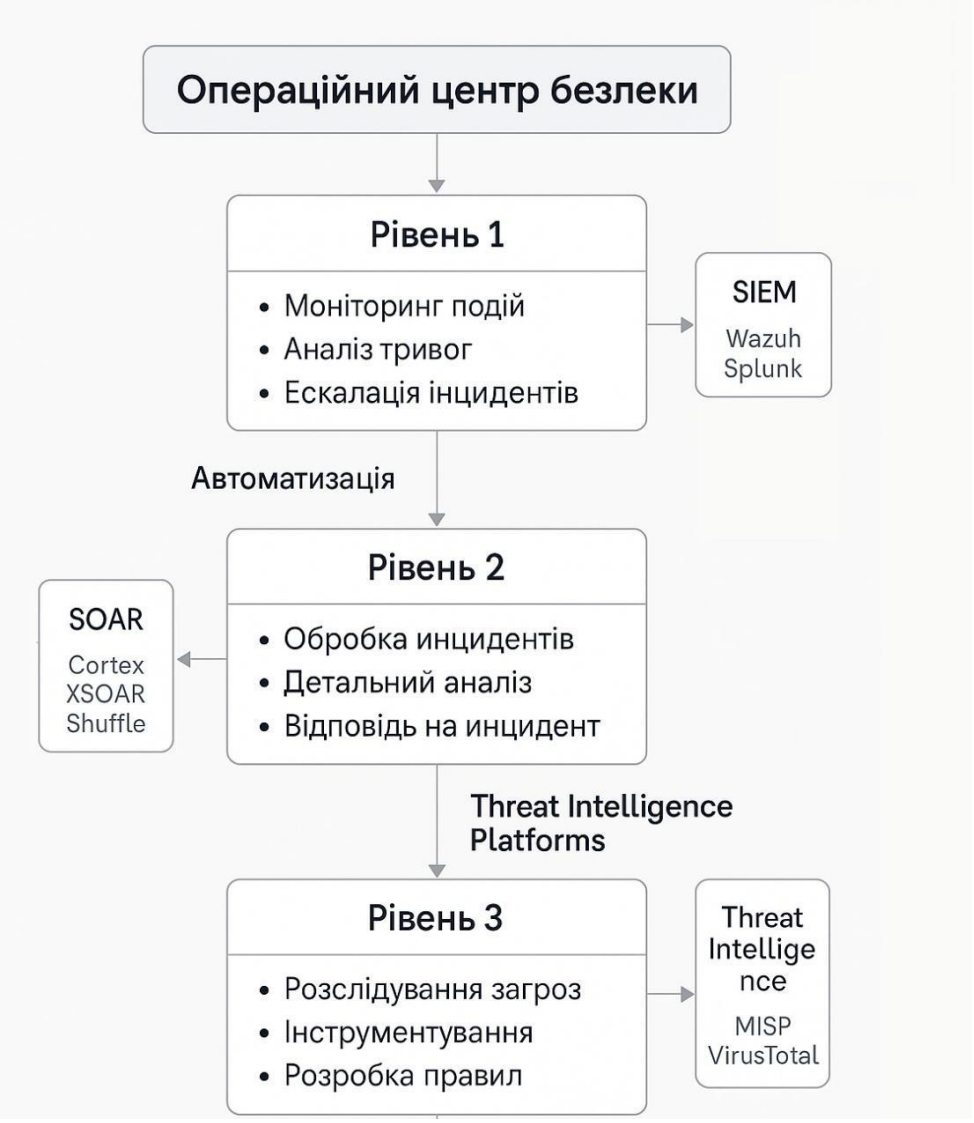


Рисунок 1.1 – Розподілення обов’язків в операційному центрі безпеки

Невеликі компанії та організації часто не можуть дозволити собі всі три рівня операційного центру безпеки, тому всю роботу виконує один невеликій відділ, через що часто можна побачити як падає ефективність захисту та час реакції на інциденти. Чим більше компанія, її інфраструктура та кількість співробітників, тим більше людей потрібно для функціонування операційного центру безпеки.

Одна з головних проблем операційного центру безпеки на сьогоднішній момент є перевантаження спеціалістів та аналітиків. Кожного дня сучасні організації генерують десятки, а то і сотні тисяч подій безпеки. Не всі події кібербезпеки становлять загрози, але не варто забувати що кожна з них потребує аналізу. Під час сильного навантаження це створює стан хронічної втоми, що тільки погіршує ефективність спеціаліста. Крім недостачу кваліфікованих кадрів, можна відмітити складність аналізу інфраструктури зростаючої компанії, без постійного розширення операційного центру безпеки. Прийняття на роботу нових співробітників, вдосконалення програмних засобів та інструментів, завжди потребує підвищеного фінансування, через що компанії стараються йти на компроміси. Наприклад використовувати штучний інтелект для звітування або інших задач, тим самим економлячи час співробітників [7].

Використання штучного інтелекту для операційного центру безпеки максимально доцільно на першому рівні, так як саме на першому рівні співробітники виконують багато мануальних та шаблонних дій, які може замінити або прискорити штучний інтелект. Це першочергова обробка інцидентів, створення поверхневих звітів, ручний запуск утиліт та знаходження інформації, формування звітів та статистики.

Стрімке зростання обсягів інцидентів інформаційної безпеки, кіберзагроз та подій зумовлює потребу у більш ефективних інструментах та підходах в роботі операційного центру безпеки. Одним із передових та найперспективніших способів підвищення якості та ефективності операційного центру безпеки є використання штучного інтелекту. Завдяки стрімкому розвитку великих мовних

моделей та їх здібностей в обробці великої кількості даних та використання додаткових інструментів, їх використання для полегшення роботи співробітників операційного центр безпеки стає необхідністю [8].

Використання штучного інтелекту можливо як у інтеграції з інструментом який використовується в операційному центрі безпеки, так і окремо від нього. Одними із прикладів інтеграції штучного інтелекту в інструменти та програмні засоби можуть бути SIEM системи, яка покращують класифікацію логів за допомогою штучного інтелекту. Або системи EDR, які дозволяють більш точно виявити підозрілу поведінку або втручання в систему на перших її етапах та прийняти максимально ефективні міри запобігання.

Використання генеративного штучного інтелекту доцільно навіть без інтеграції його всередину програмного рішення [9]. Його варто використовувати для економії часу на рутинних шаблонних задачах операційного центру безпеки, таких як заповнення таблиць, звітів, допомога у пошуку інформації з відкритих джерел. Варто зазначити, що штучний інтелект має свої недоліки та проблеми, тому при використанні його на будь якій ролі, варто спершу впевнитись в його надійності. Тестування штучного інтелекту вкрай необхідно перед його використанням, особливо якщо його використання пов’язано з вирішенням серйозних та важливих задач, де кожна помилка може спричинити серйозний удар по безпеці, фінансам або репутації компанії.

Робота присвячена автоматизації процесів першого рівня операційного центру безпеки, а саме автоматичному формуванню звітів. Розберемо детальніше роботу першого рівня операційного центру. Можемо відмінити наступні процеси операційного центру безпеки:

- реагування на інциденти кібербезпеки в реальному часі;
- аналіз даних та звітування;
- пошук інформації з відкритих джерел;
- взаємодія та передача інформації на вищі рівні операційного центру;
- комунікація з користувачами в реальному часі;

Джерелами інформації та подій на першому рівні операційного центру безпеки зазвичай стають SIEM-системи. Вони збирають інформацію з всіх необхідних джерел, таких як серверів, кінцевих пристроїв, хмарних сервісів, поштових серверів. Для збору даних на кінцевих пристроях, серверах та мережевому обладнані зазвичай налаштовуються сервіси, які в автоматичному режимі надсилають логи та події інформаційної безпеки. Для передачі цих даних використовуються різноманітні протоколи, самі популярні з них це SysLog, Netflow, Ipfix. Крім збору інформації з серверів та кінцевих пристроях, потрібно збирати та аналізувати інформацію про невдалі спроби входу в робочий акаунт чи пошту, для цього часто використовуються веб сервіси або спеціально створенні для цього системи. Невдалий вхід в робочий акаунт з різних пристроїв або IP адресів за короткий проміжок часу може означати спробу несанкціонованого доступу до облікового запису співробітника, через що інженеру операційного центру безпеки потрібно зв’язатись з власником облікового запису та прийняти всі необхідні для захисту акаунту міри. Мірами може бути заміна паролю, блокування облікового запису, або навіть повне його видалення як приклад критичної міри захисту інфраструктури.

Іноді аналітики першого рівня безпеки займаються пошуком інформації з відкритих джерел, це потрібно для отримання додаткової інформації що до інциденту. Наприклад перевірка IP адреси яка фігурує в інциденті на наявність її у публічних базах даних які показують зв’язок IP адресів з зловмисниками, або шкідливими ресурсами. Крім того за допомогою пошуку з відкритих джерел, аналітик може з’ясувати чи є адреса частиною проксі мережі, або чи використовувалась вона в будь-яких атаках до цього.

Однією з рутинних задач яку можливо автоматизувати за допомогою штучного інтелекту є звітування та ручне заповнення статистики. Для підтримки високого рівня захисту та його постійного контролю необхідне періодичне звітування, та ведення статистики. Це потрібно не тільки для спеціалістів

операційного центру безпеки, а й для керівників організації, які хочуть без занурення в складну інженерну частину оцінити якість та швидкість роботи операційного центру безпеки. В залежності від типу звіту, він може у себе включати кількість інцидентів кібербезпеки за поточний день, список найкретичніших загроз, середній час реакцію на інциденти та класифікація інцидентів за категоріями. Також штучний інтелект можна застосовувати для ведення статистики, наприклад збирати головну інформацію щоденних звітів та записувати її у файл на протязі певного періоду часу, наприклад на протязі місяця або цілий рік. Автоматизація рутиних дій не тільки дозволить отримувати більшу кількість статистики та звітів без завдання удару по фінансам компанії або часових витрат, а й підвищення швидкості реагування та аналіз інцидентів на протязі робочого дня, так як спеціалісти не відволікаються на рутинні та скучні для них завдання.

1.3. Аналіз можливостей штучного інтелекту для аналізу даних та інцидентів кібербезпеки

У сучасному світі обсяги кіберзагроз постійно зростають, а кількість даних, які необхідно аналізувати у рамках забезпечення інформаційної безпеки, перевищує можливості людини [9]. Відповідно, традиційні методи обробки інформації, побудовані на ручному аналізі логів, не встигають реагувати на реальні загрози в режимі реального часу. Саме в цьому контексті штучний інтелект (ШІ) демонструє свою ефективність як потужний інструмент підтримки та автоматизації процесів у операційних центрах безпеки.

ШІ дозволяє переходити від реактивного до проактивного захисту [10]. Замість очікування завершення атаки й подальшого аналізу, сучасні алгоритми машинного навчання та обробки природної мови можуть виявляти аномальні шаблони поведінки, передбачати можливі вектори атак та автоматично

повідомляти аналітиків про потенційні загрози ще до того, як вони завдадуть шкоди.

Одним з найважливіших напрямків є використання великих мовних моделей (LLM) у сфері безпеки. Такі моделі здатні опрацьовувати великі обсяги неструктурованих даних - логів, повідомлень SIEM-систем, повідомлень з SOC-журналів, інформаційних бюлетенів та звітів про вразливості. Вони здатні на основі цих даних сформувати повноцінний звіт, узагальнити ключову інформацію, проаналізувати типи атак, категоризувати рівні загрози та навіть надати рекомендації з усунення наслідків інцидентів.

На практиці використання ШІ в операційних центрах безпеки дозволяє значно скоротити час реагування на інциденти, зменшити кількість хибнопозитивних спрацювань, знизити навантаження на фахівців з кібербезпеки та оптимізувати процеси розслідування [9]. Завдяки навчанню на історичних даних, системи, побудовані на ШІ, можуть швидко адаптуватися до нових типів атак і виявляти складні, приховані загрози, які могли б бути пропущені традиційними засобами.

Ще однією важливою перевагою є здатність ШІ до генерації інсайтів на основі багатовимірних даних. Наприклад, за допомогою кореляції подій з різних джерел (логів операційної системи, мережевого трафіку, поведінки користувачів тощо), штучний інтелект може виявити складні ланцюги атак, де кожна окрема подія виглядає нешкідливою, але в сукупності вказує на цілеспрямовану діяльність зловмисника.

Використання ШІ в операційних центрах безпеки також відкриває нові можливості для автоматизації рутинних процесів. Наприклад, штучний інтелект може:

- самостійно генерувати виконавчі звіти для менеджменту;
- надавати технічні висновки для аналітиків рівня L1/L2;
- фільтрувати великі масиви логів за релевантністю;
- здійснювати попередню класифікацію інцидентів за MITRE ATT&CK;

- виявляти закономірності у поведінці атакувальників;
- автоматично формулювати рекомендації з реагування.

Таким чином, ШІ стає ключовим компонентом у побудові сучасної адаптивної системи кіберзахисту, яка здатна реагувати на загрози з високою точністю та швидкістю [12].

Особливої уваги заслуговують приклади інтеграції моделей GPT (Generative Pretrained Transformer) у робочі процеси операційного центру безпеки. Вони дозволяють створити інтелектуальні асистенти, які допомагають аналітикам у складанні звітів, підказують хід розслідування або навіть повністю автоматизують аналіз інциденту. Наприклад, GPT може згенерувати хронологію подій, проаналізувати типові шаблони атак або пояснити, чому певна подія класифікується як критична [9]. Завдяки цьому аналітики мають змогу зосередитись на прийнятті стратегічних рішень, замість того щоб витрачати час на рутинну обробку даних.

Штучний інтелект також дозволяє ефективно працювати з фейковими або тестовими даними у навчальних або дослідницьких середовищах. Генерація фейкових логів для симуляції інцидентів дозволяє перевірити якість алгоритмів без ризику для реального середовища. Це важливо як для навчання майбутніх фахівців, так і для тестування програмного забезпечення.

Однак, використання ШІ у кібербезпеці потребує обережності. Серед ризиків можна відмітити можливість генерації хибних висновків при недостатній якості навчальних даних, упередженість моделі або її непередбачувана поведінка. Тому важливо реалізовувати відповідні механізми перевірки, валідації та логування всіх рішень, які приймає ШІ, особливо якщо йдеться про автоматизовану реакцію на загрози.

Крім того, ШІ потребує постійного навчання та оновлення знань. Загрози постійно змінюються, а отже, моделі повинні бути адаптивними. Ідеальним варіантом є використання комбінації машинного навчання з участю людини

(human-in-the-loop), де ШІ готує первинний аналіз, а спеціаліст його коригує та підтверджує.

Загалом, впровадження ШІ у процеси аналізу подій та інцидентів в операційних центрах безпеки дозволяє підвищити якість захисту, знизити навантаження на персонал та ефективніше реагувати на сучасні кіберзагрози. Це не просто технологічний тренд, а необхідність у світі, де швидкість атаки перевищує швидкість людського реагування. ШІ дає можливість вийти на новий рівень — від реагування до прогнозування.

Важливо зазначити, що ШІ не замінює повністю людину в системах кібербезпеки, а виступає у ролі інструменту, що підсилює спроможності фахівців. Людина залишається ключовою ланкою в ухваленні рішень, зокрема в ситуаціях, коли важливо враховувати контекст або специфіку інфраструктури. Таким чином, ШІ ідеально підходить для гібридної моделі взаємодії, в якій автоматизовані інструменти допомагають людям зосередитися на складних і нестандартних завданнях.

Окремо слід розглянути використання великих мовних моделей (LLM), таких як GPT, у рамках операційного центру безпеки. Такі моделі демонструють високу ефективність у генерації звітів на основі логів, автоматизації аналізу даних та відповіді на запити в реальному часі. LLM можуть слугувати як цифрові аналітики, які швидко узагальнюють великі обсяги інформації та пропонують конкретні висновки [11]. Завдяки цьому час на обробку інцидентів скорочується, а точність рішень зростає.

Крім того, ШІ допомагає в класифікації загроз. Використовуючи алгоритми кластеризації та класифікації, система може групувати події за подібністю, виявляти нові патерни атак або виділяти відомі індикатори компрометації. Це дозволяє здійснювати превентивні заходи ще до того, як атака встигне завдати шкоди. Застосування таких алгоритмів особливо важливе в умовах APT-атак, коли зловмисники діють непомітно та протягом тривалого часу.

Насамкінець, важливим аспектом впровадження ШІ в операційному центрі безпеки є прозорість і етичність. Моделі повинні бути пояснюваними, щоб фахівці могли зрозуміти, чому система прийняла те чи інше рішення [12]. Це сприяє підвищенню довіри до технології та зменшує ризик помилкових позитивних спрацювань. Інструменти пояснюваного ШІ мають стати невід'ємною частиною таких систем.

1.4 Формалізація вимог та постановка задачі

Формалізація вимог є важливим етапом перед розробкою системи для автоматизації процесів операційного центру безпеки з використанням штучного інтелекту. Чіткі вимоги дають змогу забезпечити відповідність системи очікуванням користувачів, та отримати високу якість кінцевої розробки.

При створені засобу автоматизації операційного центру безпеки необхідно розуміти звідки система буде брати вхідні дані, яким чином вона буде їх обробляти та у якому вигляді буде зберігатись статистика та звіти, скільки ресурсів буде витрачатись на створення кожного звіту та наскільки безпечно використовувати автоматизовані сервіси на основі штучного інтелекту [14]. Базову задачею стоїть створення системи, яка повинна автоматично аналізувати інциденти інформаційної безпеки, які вона автоматично отримує з SIEM-системи та подавати результат у вигляді простого звіту, для контролю якості та ефективності операційного центру безпеки. Спеціаліст або керівник отримавши цей звіт може легко зрозуміти що відбувалось в кіберпросторі організації. Скільки було інцидентів, скільки оброблено, яка ефективність в реагуванні. Повинні бути виведені списком найкритичніші інциденти та пристрої.

Основні вимоги до програмного засобу:

– Програмний засіб повинен видавати результат у вигляді текстового файлу, а саме у форматі .txt.

- Програмний засіб повинна мати декілька варіантів звітів на вибір, наприклад виконачий звіт,технічний звіт, звіт по хронології інцидентів.
- Система повина працювати у браузеры, та бути доступною з будь якого девайсу.
- Звіти повинні мати просту та зрозумілу назву або можливість задати назву спеціалісту.
- Система має бути проста для використання, мати простий та зрозумілий інтерфейс.
- Система має підтримувати рішення вендора Wazuh, включаючи Wazuh SIEM та Wazuh Agents.

Для досягнення цілей та розробки системи, будуть проведені роботи по аналізу існуючих систем, розробка плану та тестування системи в реальних умовах.

Аналіз предметної області дає зрозуміти, що кіберзагрози представляють все більшу небезпеку. Вчасне реагування на інциденти кібербезпеки стає не просто бажаною мірою, а критично необхідною, для забезпечення достатнього рівня кіберзахисту інфраструктури компанії. Операційний центр безпеки зазвичай складається з трьох рівнів. Кожен рівень виконує свою роботу, та використовує потрібні для цього інструменти та програмні засоби. Перший рівень операційного центру робить багато мануальної роботи, що часто перевантажує його. Саме тому максимально доцільно автиоматизовувати саме перший рівень операційного центру безпеки. Автоматизація збору даних, та складання звітів є важливою та довгою роботою, яку виконують співробітники операційного центру безпеки. Автоматичний збір даних та їх аналіз, зможуть підвищити продуктивність роботи операційного центру безпеки. Тим самим підвищиться загальний рівень кібербезпеки інфраструктури. Відсутність спеціалістів, та небажання працювати на першому рівні операційного центру заставляє задуматись про автоматизацію процесів.

Програмний засіб має автоматизувати процеси збору та аналізу інформації, подаючи готовий результат у вигляді простого та зрозумілого звіту. Звіти повині бути різних видів, так як є потреба у різних варіантах аналізу даних. Робота з системою автоматизації має бути простою та зрозумілою. Головна панель управління програмним засобом знаходится у веб інтерфейсі, для доступності з будь якого пристрою. Всі файли зберігаються у базі даних, яка доступна з веб інтерфейсу, для отримання файлу потрібно лише натиснути на необхідний файл у веб інтерфейсі та завантажити його.

2 ТЕХНІЧНЕ ПРОЕКТУВАННЯ СИСТЕМИ

2.1 Розробка архітектури системи

Архітектура системи автоматизації операційного центру безпеки складається з декількох модулів, які взаємопов’язані між собою. Кожен модуль виконує свою роль та вирішує свою проблему у системі. За потреби кожен модуль можливо покращити або замінити на більш функціональний.

Основними компонентами у системі є:

- веб інтерфейс керування;
- модуль для створення аналітики;
- база даних для зберігання звітів;
- агент на кінцеві системи для збирання інформації про інциденти інформаційної безпеки та вразливості на кінцевій системі;
- платформа, яка включає в себе SIEM-систему та XDR платформу в одному програмному засобі;
- велика мовна модель;

Архітектура засобу зображена на рисунку 2.1. Користувач використовує веб-інтерфейс для вибору бажаного звіту та клікає на кнопку “Генерація”, інформація обробляється модулем створення звітів, який витягує інформацію з Wazuh SIEM та передає її у вигляді логів до великої мовної моделі. Мовна модель генерує звіт в залежності від системних та користувацьких промптів, та передає його назад до модуля створення звітів. Модуль перетворює відповідь мовної моделі у текстовий файл, зберігає звіт у базі даних та дає користувачу його завантажити. Користувач отримує бажаний звіт в текстовому форматі. Завдяки використання бази даних, користувач може завантажити звіт який був попередньо згенерований. Це економить час та ресурси, та дає змогу бачити хронологію подій.

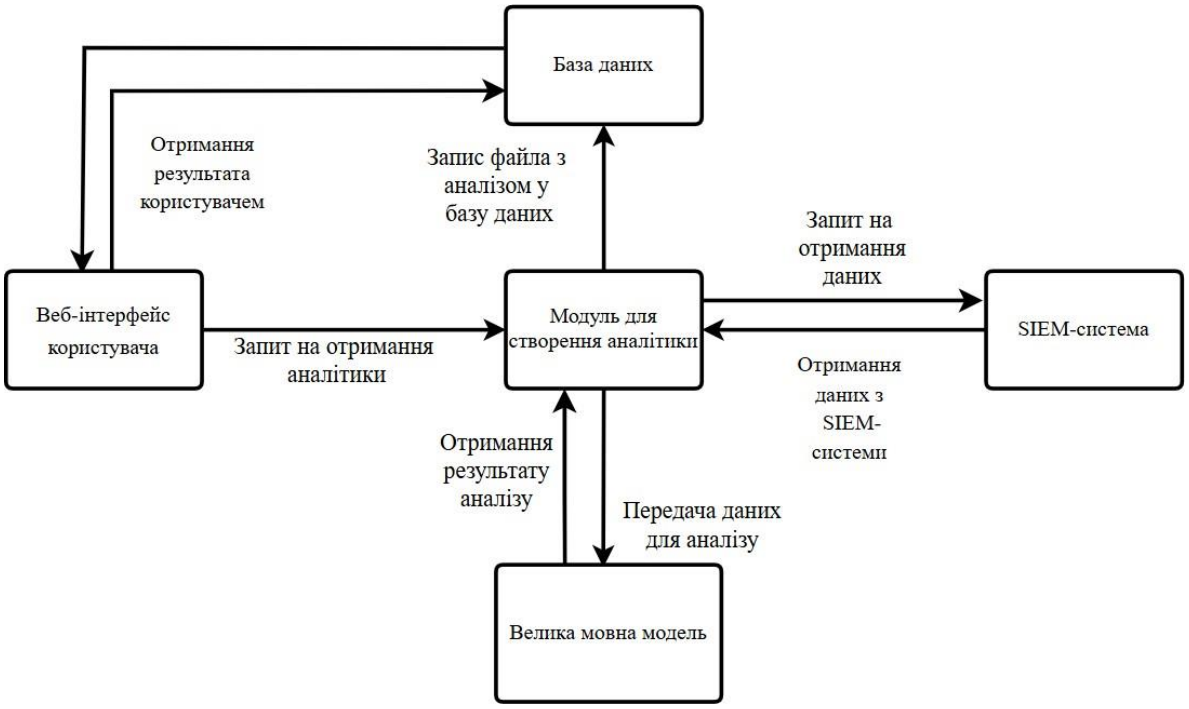


Рисунок 2.1 – Архітектура системи

Веб-інтерфейс дозволяє зручний та інтуїтивно зрозумілий спосіб взаємодії користувача з системою. Користувач може вибрати тип звіту і отримати його в текстовому файлі вже за декілька секунд. Використання веб інтерфейсу дозволяє зекономити ресурси та пам’ять комп’ютера в порівнянні з встановленням десктопного застосунку. Крім зручності використання, звіт можна завантажити з будь якого пристрою у якого є доступ до браузеру.

Модуль для створення аналітики отримує інформацію з SIEM використовуючи внутрішні API, передає їх великі мовній моделі для генерування простого та зрозумілого звіту, після чого записує готовий звіт у базу даних. У вигляді штучного інтелекту модуль може використовувати будь яку популярну модель, будь вона локальною або онлайн.

База даних забезпечує зберігання звіту на довгий період, що дозволяє отримати вже згенерований звіт без повторної його генерації. База даних зберігає звіти у текстових файлах, в точно такому ж форматі як і оригінальні звіти. Це зроблено для швидкого отримання звітів з бази даних без попереднього форматування звіту на сервері.

Велика мовна модель використовує підготовлені розробником промпти для генерації різних звітів на основі отриманих логів. Штучний інтелект має великий потенціал у структуруванні інцидентів інформаційної безпеки та створення звітів. Його правильне використання часто дає кращий результат ніж модулі, які структурують події по чітко визначеним правилам. Крім того, завдяки генерації тексту у звітах мовними моделями, навіть не технічний користувач може зрозуміти технічні аспекти роботи операційного центру безпеки. Це чудово підходить для керівників або консультантів, які мають бажання слідкувати за виконаною роботою, але не мають технічних навичок.

Кожен компонент може хоститись як на окремій віртуальній машині, так і на одній і тій ж самій. Система модульна, тому ми з легкістю можемо розвернути її компонента на окремих віртуальних машинах, або навіть використати docker deployment для деплоя кожного окремого компоненту у своєму docker контейнері. При виборі варіанту деплоя кожного компоненту, потрібно розраховувати кількість ресурсів які готові виділитись під систему, та умови до відмовостійкості. Наприклад, при використанні контенеризації, у випадку виходу з ладу системи на якій встановлюється docker, з ладу може вийти вся система. Для повної відмовостійкості можливо встановлювати кожен компонент на окремій віртуальній машині, а також мати можливість швидко замінити у випадку виходу з ладу одного або декількох компонентів.

Компоненти розраховані як на OS Windows, так і на Linux. Це гарантує безпроблемне впровадження на будь яку інфраструктуру. Завдяки простоті впровадження та простого виправоення проблем при роботі, спеціалістам операційного центру або айті департаменту не знадобиться витрачати багато часу або грошей на впровадження даної системи. Адже швидкість впровадження, час на виправлення помилки у програмних засобах та інструментах грає важливу роль при виборі програмного засобу або цифрової системи для операційного центру.

2.2 Модуль для створення аналітики

Модуль для створення аналітики є центральною частиною засобу. Він відповідає за отримання всіх необхідних даних з SIEM-системи, передачі даних та промпту до штучного інтелекту, запис готового звіту до бази даних. Компонент повинен швидко комунікувати з іншими модулями в системі для швидкої роботи без збоїв та помилок. Схема взаємодії SIEM-системи та джерела даних зображено на рисунку 2.2.

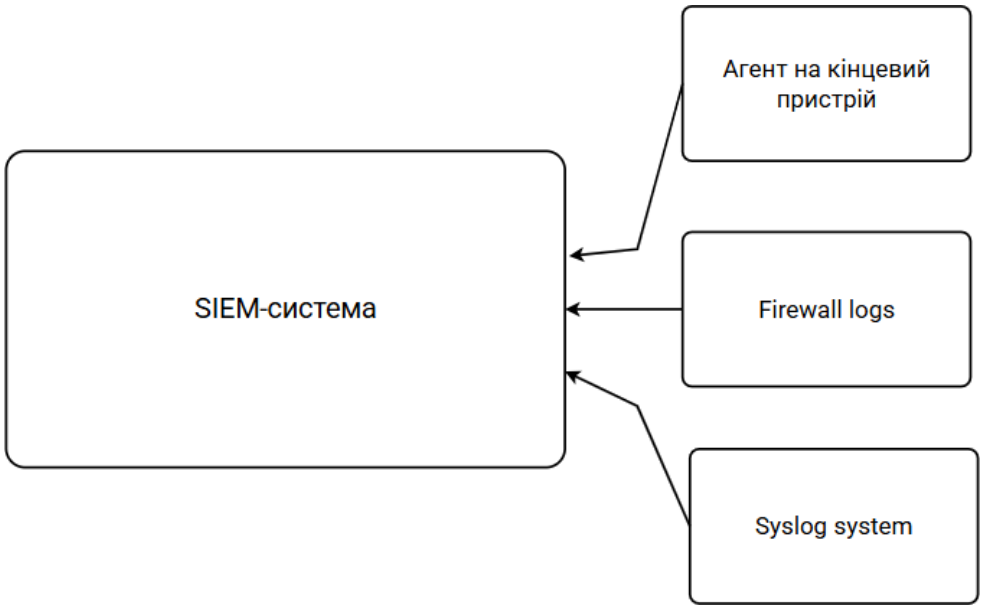


Рисунок 2.2 – Схема взаємодії SIEM-системи з джерелом даних

Для генерування звітів використовується генеративний штучний інтелект, який на вході отримує промпт з описом звіту та інформацію щодо інцидентів інформаційної безпеки у вигляді логів з SIEM-системи. В залежності від промпту, штучний інтелект видає звіт в певному форматі, та з певними даними. Промт зберігається у вигляді тексту для простого редагування на серверній частині застосунку. Кожен варіант звіту передбачає собою варіант свій варіант промпту, та свій варіант логів.

Для генерації звітів які відповідають шаблонам, або іншим бажаним параметрам, використовується так звані системні промпти. Вони дозволяють зазадалегідь задати правила що до відповіді мовної моделі. Таким чином кожен промпт буде невеликої довжини, або навіть декілька символів для зручності. А вся інструкція що до генерування звітів та аналізу даних буде створена та налагодженна зазадалегідь за допомогою system-повідомлень для API ключів або чатів.

Для отримання правильних та точних результатів, необхідно використовувати правильно написані промпти. Цього добились шляхом використання спеціальних технік промпт інженерії, які допомагають підвищити точність отриманого результату та його вміст. У роботі були використані наступні техніки промпт інженерії:

1. Рольова інструкція. Ця техніка передбачає задання ролі штучному інтелекту, наприклад: «ти аналітик безпеки SOC з багаторічним стажем, який чітко вміє розрізняти загрози та логи в системі Wazuh.»
2. Приклад запиту. Ця техніка передбачає надання штучному інтелекту прикладу запиту, що полегшує його роботу в розумінні інформації.
3. Контекстне уточнення. Іноді необхідно уточнити деякі критичні моменти, для точної роботи штучного інтелекту. Наприклад яке саме поле в JSON файлі відповідає за рівень загроз. Або що рівень 7 та вище у полі загрозі варто сприймати як критичний.
4. Спосіб думок. Варто дати штучному інтелекту послідовність думок, ніби ми вчимо людину. Це потрібно для того, що б штучний інтелект не придумував кожен раз новий спосіб вирішення завдання, а чітко йшов по заданим крокам. Наприклад, спочатку знайди всі логи з рівнем загроз вище ніж 5, потім зроби дію1.
5. Стиль відповіді. Для отримання відповіді в бажаному стилі, варто зазначити бажаний стиль у промпті. Наприклад, дай відповідь використовуючи

технічні терміни, але зроби пояснення ніби для п’яти річної дитини, щоб будь хто зміг зрозуміти як все відбувається.

6. Кінцевий результат. Для отримання результатів в майже однаковому форматі, потрібно чітко задати кінцевий результат. Скільки слів, речень, який об’єм файлу, який тип файлу, чи дозволене використання таблиць, малюнків.

Точність у отриманих звітах є ключовою якістю у використанні системи автоматизації. Кожен користувацький промпт та системний промпт по своєму впливає на отриманий результат. Перед використанням нового промпту або system-промпту необхідно протестувати кожен з них на різних наборах даних. У випадку неточності, варто відредагувати інструкцію для штучного інтелекту. Також, необхідно враховувати що різні моделі по різному виконують одні і ті ж інструкції, тому при зміні або оновленні моделі, варто знову провести повний цикл тестувань перед впровадженням системи.

2.3. Алгоритм роботи системи

Для виконання поставлених задач вкрай необхідно мати алгоритм роботи системи. Кожен компонент системи виконує свою задачу, правильне їх компонування забезпечує отримання бажаного результату. Для створення алгоритму роботи програмного застосунку, було враховано можливості винекнення помилок та збоїв при роботі програми. Варто зазначити, що алгоритм роботи програмного застосунку підходить для великої кількості SIEM-систем, та великих мовних моделей. У випадку, якщо при використанні системи виникне потреба у використанні іншої мовної моделі, будь вона локальна чи онлайн, модливо легко перевести застовунок на нову велику мовну модель, без втрати функціоналу.

Алгоритм роботи програми зображений на рисунку 2.3.

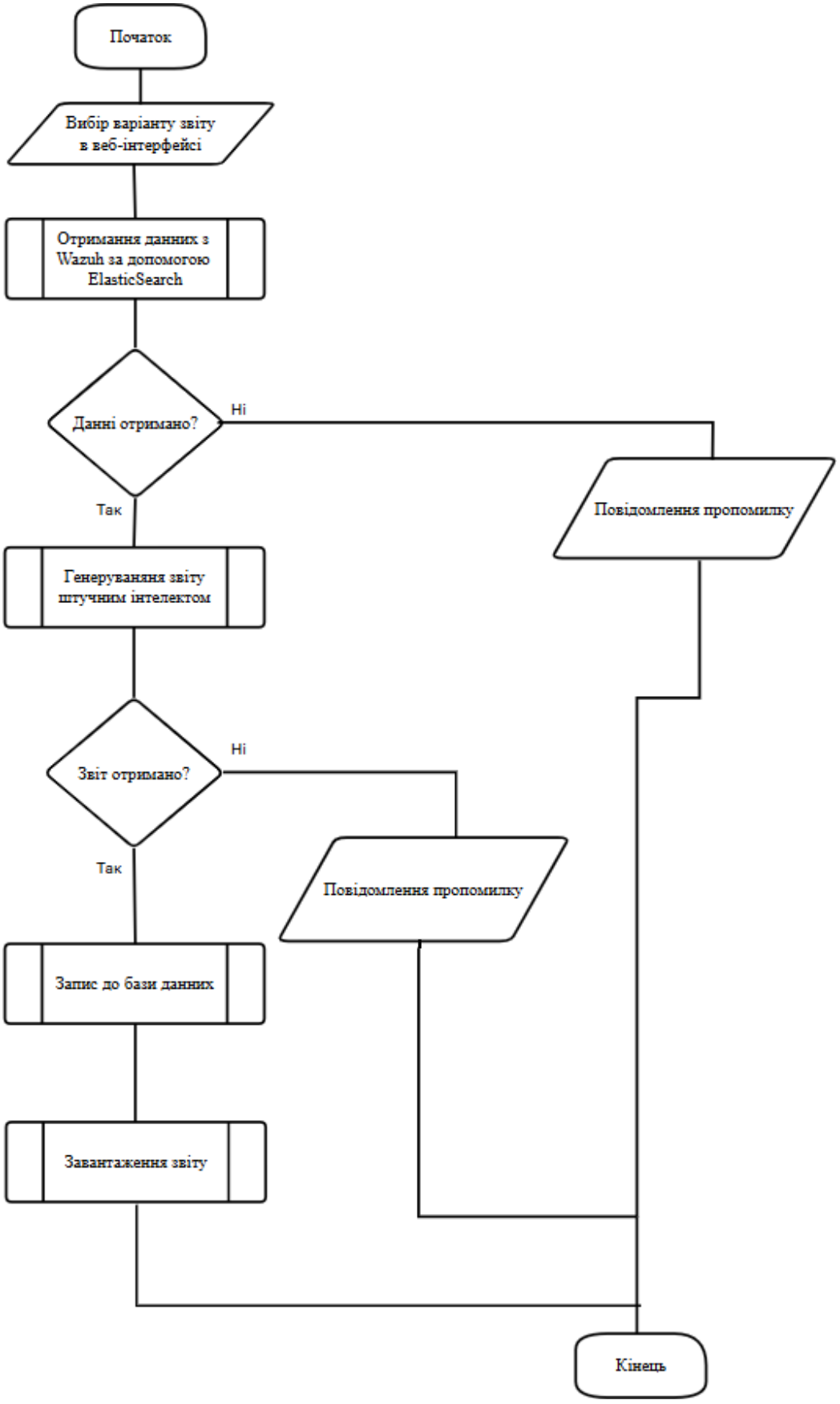


Рисунок 2.3 – Алгоритм роботи програмного застосунку

Опис алгоритму програми:

1. Користувач вибирає тип звіту в веб-інтерфейсі, веб-інтерфейс поводить себе як клієнтська частина, задача якої отримати дані від користувача та передати їх на серверну частину.
2. Модуль створення аналітики спочатку має отримати данні про інциденти інформаційної безпеки з SIEM-системи. В залежності від типу звіту, модуль отримує різні дані за різний період часу.
3. Якщо дані не отримано, користувач бачить повідомлення про помилку та програма завершує своє виконання.
4. У випадку успішного отримання даних модуль робить запит до штучного інтелекту, з використанням API ключа. Штучний інтелект генерує звіт по заданим в користувацькому промпті та системному промпті даним, та передає його в текстовому форматі назад до модуля створення звітів.
5. Якщо сталась помилка та штучний інтелект не вернув дані, наприклад у нас вийшов час для використання API ключа, то користувач отримує помилку а програма завершує своє виконання.
6. У випадку успішного отримання відповіді від штучного інтелекту, модуль створення звітів перетворює текстову відповідь у окремий текстовий файл, та записує його до бази даних.
7. Після чого користувач завантажує звіт у текстовому форматі на свій комп’ютер через браузер.

В алгоритмі варто враховувати помилки при взаємодії компонентів системи. У випадку неполадок або системних помилок, користувач має бачити повідомлення про помилку, а також короткий її опис. У випадку знехтування цим правилом, при неполадках у системі користувач може не зрозуміти, що сталась помилка, а думати що система просто довго обробляє його запит, або ж у нього проблеми зі з’єднанням. Помилки можуть бути нестандартними, або виникнути між компонентами між якими по задумці вони виникнути не могли. Тому під час

створення алгоритму варто виявляти помилки, або неправильне виконання програмного засобу на кожному його кроці, будь то отримання даних або робота модуля створення звітів.

У розділі було розглянуто архітектуру системи автоматизації створення звітів для операційного центру безпеки за допомогою штучного інтелекту. Архітектура системи модульна, де кожен елемент виконує свою роль та вирішує свою проблему. Систему можна впроваджувати як на декількох віртуальних машинах, дотримуючись модульності, так і на одні для економії ресурсів та часу. Наведено опис кожного елементу системи та спосіб його комунікації з іншими елементами системи. Система може бути впроваджена як на Windows системах, так і на Linux. Також можливе використання контенеризації для швидкого встановлення всіх компонентів в одному середовищі.

Було наведено сценарії на випадок помилки в процесі роботи алгоритму. Алгоритм роботи програмного засобу було створено та наведено як в схематичному вигляді так і в текстовому для повного його розуміння.

3. РОБОЧЕ ПРОЕКТУВАННЯ СИСТЕМИ

3.1. Обґрунтування вибору інструментальних засобів розробки

Для розробки засобу автоматизації процесів операційного центру варто приділити особливу увагу до інструментів та технологій які будуть використовуватись [13]. Правильне створення програмного застосунку дозволить не тільки підвищити ефективність операційного центру безпеки, а й зекономити час на підтримку програмного застосунку та виправлення помилок, багів в його роботі. Готовий програмний засіб має автоматично аналізувати дані за певний період часу, та видавати їх результат у зручному для користувача форматі.

У процесі розробки програмного засобу для автоматизованого аналізу даних та інцидентів кібербезпеки було розглянуто кілька сучасних мовних моделей, серед яких Gemini (від Google), Claude (від Anthropic), DeepSeek (від DeepSeek AI) та OpenAI GPT. Кожна з цих моделей має свої переваги, проте остаточний вибір було зроблено на користь OpenAI GPT з кількох ключових причин.

По-перше, OpenAI GPT є найбільш стабільною, зрілою та широко використовуваною моделлю з потужним API, що активно підтримується розробниками по всьому світу. Компанія OpenAI має значний досвід у дослідженнях штучного інтелекту, а її продукти проходять ретельне тестування в різних галузях, включно з кібербезпекою, автоматизацією бізнес-процесів і аналізом великих обсягів тексту.

По-друге, GPT-моделі, зокрема GPT-4o, демонструють найвищу якість генерації тексту серед відкритих моделей, включаючи глибоке розуміння контексту, здатність до логічного аналізу, та послідовну структуру відповідей. Це критично важливо для завдань, пов’язаних із генерацією звітів про інциденти безпеки, де важлива точність, узгодженість та аналітичність викладу.

По-третє, платформа OpenAI має добре задокументований API, який легко інтегрується у Python-застосунки, що спростило розробку власного інструменту.

На відміну від деяких інших моделей, доступ до GPT API не вимагає складної інфраструктури — це дозволяє запускати продукт навіть на локальному рівні з мінімальними вимогами.

Для аналізування та підготовки звіту буде використовуватись штучний інтелект, а саме мовна модель від OpenAI. Серед всіх існуючих моделей від OpenAI було обрано модель GPT-4o. Таблиця 3.1 предсавляє собою таблицю порівняння великих мовних моделей від OpenAI.

Таблиця 3.1 – Порівняння велиуих мовних моделей від OpenAI

Параметр	GPT-3.5	GPT-4	GPT-4o
Архітектура	Трансформер (адаптована версія GPT-3)	Модифікований трансформер (багатомодельна архітектура)	Уніфікована мультимодальна модель (текст, зображення, аудіо)
Максимальна довжина контексту	4096 токенів	8192 токенів (до 128k у розширеній версії)	128000 токенів
Середня латентність відповіді (сек)	0.7–1.2 сек	2–4 сек	0.3–0.7 сек
Пропускна здатність (токенів/сек)	~60–80 токенів/сек	~30–40 токенів/сек	~150–200 токенів/сек
Підтримка мультимодальності	Ні	Обмежена (тільки GPT-4-vision)	Так (вбудовано)
Оптимізована для API	Так	Так	Так
Сумісність з SOC автоматизацією	Обмежена (не найкраща для великих log-масивів)	Висока точність аналізу, але повільна генерація	Оптимальна для реального часу та великих обсягів логів

GPT-4o представляє собою онлайн модель, доступ до якої, я буду отримувати за допомогою API ключа. Використання API OpenAI у рамках розробки програмного засобу для автоматизації аналізу кіберінцидентів є стратегічно обґрунтованим рішенням, яке забезпечує значні переваги над звичайним

вебінтерфейсом ChatGPT. Насамперед, API дозволяє повністю автоматизувати робочий процес. Якщо вебінтерфейс орієнтований на ручне введення запитів користувачем, то API забезпечує можливість програмного надсилання запитів до мовної моделі, обробки відповіді та інтеграції її у внутрішні бізнес-процеси.

Другою важливою перевагою є можливість інтеграції API у будь-яку інфраструктуру. Це означає, що модель можна підключити до систем збору логів, таких як Wazuh SIEM, до бази даних, до внутрішнього вебінтерфейсу організації чи до системи сповіщення. Така інтеграція значно розширює можливості використання моделі і дозволяє використовувати її не як ізольований інструмент, а як повноцінний компонент у складі складної автоматизованої системи.

API також дає гнучкість у керуванні параметрами генерації тексту. Розробник може точно вказати бажану температуру моделі, обмеження на довжину відповіді, налаштувати системні інструкції та історію попередніх повідомлень. Це критично важливо для досягнення передбачуваності, стабільності і точності результатів. У випадку з вебінтерфейсом, користувач обмежений у можливості зміни таких параметрів.

Важливим фактором є масштабованість [14]. За допомогою API можна запускати генерацію звітів автоматично за розкладом, у відповідь на певні події або в рамках цілодобової обробки потоків даних. Це дозволяє використовувати мовну модель не лише для одиничних запитів, а для системного аналізу великих масивів інформації.

Окрему увагу варто звернути на питання безпеки. При використанні API доступ до моделі захищений унікальним ключем, що дозволяє забезпечити контроль доступу та відповідність політикам безпеки в корпоративному середовищі [15]. Крім того, API дозволяє зберігати результати генерації у структурованому вигляді, що спрощує ведення аудиту та аналізу.

OpenAI також має власні бібліотеки для інтеграції штучного інтелекту в програмні засоби, та зручного використання API ключа. За допомогою API ключа,

я зможу легко інтегрувати використання моделі в свій програмний засіб без потреби створювати складні модулі інтеграції.

Вибір SIEM-системи як основного джерела даних для аналізу в межах цієї роботи є цілком обґрунтованим і логічним. SIEM (Security Information and Event Management) - це платформа, яка забезпечує централізований збір, зберігання, кореляцію та аналіз логів і подій з різних систем безпеки, серверів, мережевого обладнання, застосунків та інших джерел [16]. Саме завдяки своїй комплексності SIEM стає ідеальним кандидатом для інтеграції з системою автоматизованого аналізу інцидентів .

Насамперед, SIEM-система акумулює всю ключову інформацію про події, що відбуваються в мережі організації. Вона дозволяє отримати структуровані дані про інциденти: IP-адреси джерел атак, часові мітки, рівні загроз, типи атак, назви хостів та інші важливі атрибути. Це забезпечує якісний та достатній контекст для аналізу, необхідний для генерації як технічних, так і виконавчих звітів.

Крім того, використання SIEM дозволяє уникнути хаотичного збору логів з окремих джерел. Натомість надається єдиний уніфікований потік подій, що спрощує інтеграцію з мовними моделями та іншими аналітичними системами. Такий підхід забезпечує сталість формату даних, що критично важливо для точного аналізу та роботи з ШІ.

Під час вибору SIEM-системи потрібно було опиратись на наступні якості:

- безкоштовне використання або open source;
- можливість збирати логи з багатьох джерел;
- можливість використання агента на кінцевий пристрій, який легко інтегруєтсья з SIEM-системою;
- використання внутрішнього API для отримання даних;

Вибір впав на систему Wazuh. Вона має простий та зрозумілий інтерфейс, інструкцію та документацію по його використанню. Система безкоштовна та має підтримку та оновлення. Також виробник має агент на кінцевий пристрій з назвою Wazuh Agent, який швидко свтановлюється на Windows та Linux системи, та

швидко інтегрується з SIEM-системою Wazuh. Крім встановлення на Windows чи Linux, система може бути встановленною в контейнері, наприклад Docker.

Для простого встановлення SIEM-системи яка складаються з декількох компонентів, було обрано ти встановлення за допомогою контейнеризації та

для простого та швидкого запуску всіх контейнерів одночасно, та централізовано змінювати їх параметри.

Для головного модуля створення звітів було використано мову програмування Python. Вибір саме на Python впав по декільком причинам. По перше, мова програмування Python має велику кількість навчальних ресурсів та курсів, що дозволяє легко знайти вирішення проблеми у процесі розробки [17]. По друге, мова програмування Python має велику кількість бібліотек та фреймворків, що сильно полегшує створення програмних застосунків мовою Python.

Для роботи з API ключем від OpenAI потрібно використовувати необхідні для цього бібліотеки. Вибір впав на бібліотеку openai від компанії OpenAI. Бібліотека openai для Python забезпечує простий та надійний інтерфейс для взаємодії з моделями від OpenAI, в особливості з GPT-4o. Бібліотека дозволяє у пару рядків створювати необхідні запити, та отримувати відповіді на них. Переваги цієї бібліотеки у офіційній підтримці та стабільності [17].

Для програмування серверної частини веб інтерфейсу було обрано мову програмування Python. Мова була обрана через свою простоту, читабельність та велику кількість бібліотек та фреймворків. Веб сервер було реалізовано за допомогою фреймворку Flask, який дозволяє швидко та легко будувати серверну частину, та інтегрується з HTML інтерфейсами.

Для клієнтської частини веб інтерфейсу було обрано базові технології, такі як HTML, CSS та JavaScript. Ці базові технології вже довгий час є невід’ємною частиною клієнтської сторони всіх веб додатків. Також JavaScript використовується для взаємодії з серверною частиною та динамічного оновлення сторінки.

Для зберігання аналітики та звітів було зроблено систему зберігання даних на основі SQLite. Це просто реляційна база даних, яка не потребує окремого сервера та ідеально підходить для локальних додатків.

Для написання та редагування коду було використано редактор коду Visual Studio Light. Це легкий та кросплатформений редактор коду, з його головних переваг можна виділити безкоштовне використання, підтримку великої кількості мов програмування, велику кількість розширень та плаганів. Visual Studio Light є лідером в категорії редактору коду для розробки веб додатків завдяки своїй простоті та доступності на майже всіх операційних системах.

3.2. Програмна реалізація застосунку

Для реалізації застосунку автоматизації процесів операційного центру безпеки, було виконано наступні задачі:

1. Встановлено SIEM-систему Wazuh.
2. Встановлено агенти на кінцеві пристрої для збору інформації про системні інциденти.
3. Створено веб додаток для створення аналітики через веб браузер з будь якого пристрою.
4. Налаштовано модуль для використання штучного інтелекту для аналізу даних.

Для встановлення SIEM-системи Wazuh було обрано систему на базі Linux, Parrot OS. Система Wazuh має декілька способів розгортання. Серед них розгортання на віртуальній машині, розрортання за допомогою Docker Compose, розгортання за допомогою Puppet, Ansible та інші. Варіативність варінту встановлення дає змогу впроваджувати систему використовуючи максимально зручний для інфраструктури варіант. Варіанти встановлення Wazuh показані на рисунку 3.1.

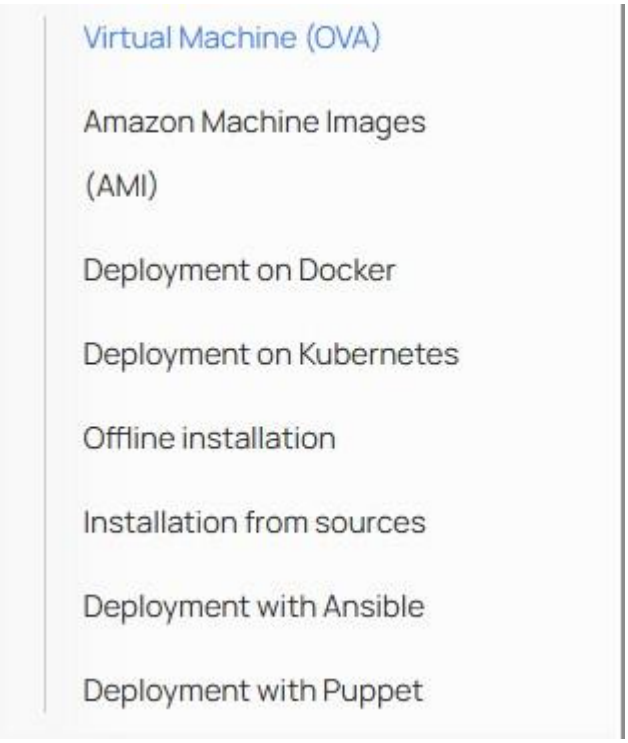


Рисунок 3.1 – Способи розгортання Wazuh з офіційного сайту

Серед багатьох способів розгортання я обрав спосіб з Docker контейнерами. Так як SIEM система складається з трьох компонентів, Wazuh indexer, Wazuh manager та Wazuh dashboard. Потрібно було розгорнути їх всі на одній системі, було обрано спосіб з використанням Docker Compose. Цей спосіб дозволяє централізовано запускати всі три контейнера та слідкувати за їх роботою, крім того всі три компоненти повині бути зв'язанні між собою. Використання Docker Compose дозволяє не плутатись в зв’язувані трьох контейнерів між собою, а робить це автоматично.

Для запуску SIEM-системи нам потрібно перейти в директорію з файлом docker-compose.yml та ввести команду sudo docker-compose up -d. Після чого всі 3 контейнера піднімуться одночасно, та система буде готова до першопочаткового використання.

У ролі системи для агента на кінцевий пристрій я вибрав систему на базі Linux, Kali Linux. На нього був встановлений агент на кінцевий пристрій від виробника Wazuh SIEM, Wazuh Agent. Після Встановлення агента на кінцевий

пристрій, я з’єднав агент та SIEM систему за допомогою внутрішньої команди для конекту агента та SIEM-системи.

Головною складовою програмного засобу для автоматизації процесів операційного центру безпеки можна вважати штучний інтелект та модуль для його використання [18]. Робота великої мовної моделі openai неможлива без промптів. Промпти потрібно задавати правильно, з використанням сучасних технік промпт інжинірінгу. Правильно заданий промпт та системний промпт прямо впливає на результат та точність роботи застосунку.

Для початку розберемо використання промптів в бібліотеці openai на прикладі модуля створення аналітики. Та використання команд в бібліотеці OpenAI в модулі створення аналітики.

Для кожного запиту бібліотека дає нам 2 вида промптів. Системний промпт та користувацький промпт.

```
    messages = [
            {"role": "system", "content": (
                "Ти — аналітик кібербезпеки SOC рівня L1. "
                "Твоя задача — проаналізувати логи з SIEM-системи Wazuh, які надходять у форматі JSON. "
                "Вхідні дані містять такі поля: @timestamp (час події), rule.description (опис інциденту), "
                "rule.level (рівень загрози від 0 до 15), agent.name (назва хоста), full_log (повне повідомлення), "
                "location (джерело логів), mitre (MITRE ATT&CK), data.dstuser, data.srcip тощо. "
                "Якщо level >= 7 — вважай це критичною подією. "
                "На основі логів згенеруй відповідь у залежності від типу звіту. "
                "Пиши українською мовою. Структурованість, узгодженість і логіка — обов’язкові. "
                "Відповідь має бути у вигляді аналітичного тексту, зрозумілого для людини. "
                "Уникай шаблонних фраз, поясню контекст."
            )},
```

```
        {"role": "user", "content": f"{prompt}\n\n{alerts_data}"}
    ]
```

Системний промпт позначено як “role”: “system”. Користуватський позначено як “role”:”user”. При написанні системного промпту, наша задача чітко визначити ролі штучного інтелекту та його задачу. Визначити у якому форматі він отримає запит, та у якому форматі він має відповісти.

Розберемо системний промпт для нашого застосунку:

"Ти — аналітик кібербезпеки SOC рівня L1. " - Тут чітко задаємо роль для штучного інтелекту. В даному випадку штучний інтелект буде чітко розуміти свою роль та підточувати відповіді під неї.

"Твоя задача — проаналізувати логи з SIEM-системи Wazuh, які надходять у форматі JSON. " – Тут даємо задачу, та показуємо подробиці користувацького запиту. Для прикладу що логи будуть йти у форматі JSON.

"Вхідні дані містять такі поля: @timestamp (час події), rule.description (опис інциденту), rule.level (рівень загрози від 0 до 15), agent.name (назва хоста), full_log (повне повідомлення), location (джерело логів), mitre (MITRE ATT&CK), data.dstuser, data.srcip тощо. " – Використовуємо уточнення для того, щоб штучний інтелект розумів нюанси, а не вигадував їх при кожному запиті, що могло б спричинити неточні результати.

"Якщо level >= 7 — вважай це критичною подією. " – даємо спосіб думок.

"На основі логів згенеруй відповідь у залежності від типу звіту. Пиши українською мовою. Структурованість, узгодженість і логіка — обов’язкові. Відповідь має бути у вигляді аналітичного тексту, зрозумілого для людини.

"Уникай шаблонних фраз, поясню контекст." – додаємо уточнення до задачі.

Крім системних промптів необхідно використовувати користувацькі промпти, або як їх ще називають user-prompts. Для кожного репорту ми використовуємо свій промпт. Пропоную розібрати три користувацькі промпти які ми використовуємо для формування аналітики та репортів. Промпт для сумарного

аналізу: «Створи короткий звіт по системним подіям. Звіт має наступну структуру: 2–4 речення висновку простими словами. Наприклад, було виявлено 22 інцидента, серед яких 5 критичних, 3 хоста мають критичні інциденти. Далі наведи списком: Топ 5 категорій інцидентів, топ 5 категорій критичних інцидентів, топ 5 хостів по інцидентам. Не потрібно давати рекомендацій, просто аналіз даних. В кінці напиши /haho/ для того щоб я бачив що ти зрозумів промпти». Спочатку ми чітко задаємо структуру, без цього кожен звіт буде мати різний вигляд, що не правильно при аналізі подій операційного центру безпеки. Коли звіти мають однакову структуру, їх легше читати та орієнтуватись в контенті звіту. Крім того чітка структура полегшує роботу штучному інтелекту, та запобігає багатьох неточностей. Після цього додаємо конкретні приклади, щоб штучний інтелект бачив який результат ми від нього очікуємо. Для уточнення ставимо чіткі рамки, наприклад, що не потрібно давати рекомендації що до усунення інцидентів або використовувати інші речі всередині звіту. Таким чином штучний інтелект буде дотримуватись певних рамок, що дозволяє отримувати результат аналізу у чітко структуровані формі. В кінці ми добавили просту перевірку на правильність отримання штучним інтелектом промпту: «В кінці напиши /haho/ для того щоб я бачив що ти зрозумів промпти.». Таким чином в кінці кожного файлу з аналізом даних, ми будемо отримувати простий рядок з символами «/haho/». Це дуже зручно для того щоб виявити помилки при отримані промпту, адже у випадку неправильнох з будь яких причин роботи програми або штучного інтекту, ми будемо отримувати не правильні результати аналізу. Самий простий спосіб виявити помилку це виключіти можливі фактори, наприклад що штучний інтелект не отримав користувацький промпт, а видав результат тільки використовуючи дані з SIEM-системи, та системний промпт.

Промпт для технічного аналізу: «Проаналізуй події технічно. Спочатку зроби короткий вступ — 2–3 речення про загальну кількість подій, кількість критичних подій (рівень ≥7) та кількість залучених хостів. Далі побудуй структурований список інцидентів. Для кожного інциденту вкажи: короткий опис, rule.description,

тип загрози або атаки (якщо є), IP-адресу джерела, користувача (data.dstuser), назву хоста (agent.name), рівень загрози (rule.level), час події (@timestamp). Обов'язково познач критичні інциденти окремо. Пиши аналітичним стилем, українською мовою. Уникай шаблонності. В кінці додай тег /haho/ для валідації промпту.». Цей промпт орієнтований на технічних спеціалістів, які аналізують інциденти безпеки. Він вимагає спершу короткий підсумок усієї ситуації, потім розбір кожного інциденту в деталях, включаючи ключові поля логів. Особливу увагу модель повинна приділити критичним подіям (з рівнем ≥7). Тут ми використовуємо ті самі техніки промпт інженерії, для правильної роботи програми.

Промпт для хронології інцидентів: «Створи структурований хронологічний звіт подій безпеки. Почни з короткого вступу: загальна кількість інцидентів, діапазон часу, кількість хостів, на яких фіксувались події. Далі представ хронологію інцидентів у порядку зростання часу. Для кожного інциденту вказуй: час події (@timestamp), короткий опис, rule.description, IP-адресу джерела, хост (agent.name), користувача (data.dstuser), рівень загрози (rule.level). Виділяй ключові зміни поведінки атакуючих (наприклад, ескалацію привілеїв, зміну напрямку атак або підозрілу активність). В кінці підведи підсумок: які тренди спостерігались, які типи атак домінували, на які хости слід звернути особливу увагу. Пиши аналітично, стисло та зрозуміло для спеціалістів SOC. Завершуй тегом /haho/.». Цей промпт дозволяє побудувати логічний ланцюжок подій у часі. Він ідеально підходить для ситуацій, коли потрібно проаналізувати розвиток атаки чи серії інцидентів. Спершу модель генерує вступну частину, потім перелік подій у хронологічному порядку. Особливу увагу штучний інтелект має звертати на зміни поведінки атакуючих, це важливий маркер у реальному інцидент-менеджменті. У кінці додається загальний підсумок і ключові висновки.

Для того щоб отримати аналіз даних, недостатньо просто написати промпти, необхідні функції які зможуть отримати дані з SIEM системи, передати їх штучному інтелекту, та створити текстовий файл з аналізом потрібних для нас даних.

Розберемо футкцію `generate_reports()`. Задача функції у тому, що б спочатку отримати дані з SIEM-системи, передати дані на штучний інтелект, та зберегти звіт у базу даних. Подивимось на частину кода яка відповідає за отримання даних з SIEM-системи.

```
wazuh_response = requests.get(
        WAZUH_URL,
        auth=(WAZUH_USER, WAZUH_PASS),
        headers={"Content-Type": "application/json"},
        json={
            "size": 50,
            "sort": [{"@timestamp": {"order": "desc"}}]
        },
        verify=False
    )
```

Розберемо його, `WAZUH URL, WAZUH_USER, WAZUH_PASS` – ці дані зберігаються у файлі .env, для того щоб не записувати чутливу інформацію в головний файл. Задача коду аутентифікувати юзера за допомогою паролю та іменем користувача, після чого отримати логи певної довжини.

Розглянемо шматок коду, який перевіряє отримання даних та видає помилку у разі невдачі.

```
    if wazuh_response.status_code != 200:
        return jsonify({"error": "Не вдалося отримати дані з Wazuh"}),
500
```

Це потрібно для того, щоб у разі проблемою з паролем, іменем користувача, або SIEM-системою, ми могли чітко помітити що проблема саме на стороні отримання даних з SIEM-системи.

Для вибору мовної моделі, ми в коді чітко маємо зазначити, яку мовну модель ми збираємось використовувати.

```
completion = client.chat.completions.create(
        model="gpt-4o",
        messages=messages
    )
```

В даній частині коду ми виберажмо використовувати саме gpt-4o, згідно з нашою таблицею порівняння 3.1.

Після успішного отримання даних їх потрібно записати до бази даних, та перевести перед цим у текстовий формат.

```
  with open(path, "w", encoding="utf-8") as f:
        f.write(result_text)
```

У даній частині коду, ми виконуємо запис отриманої від штучного інтелекту відповіді у текстовий файл.

```
with sqlite3.connect(DB_NAME) as conn:
        conn.execute("INSERT INTO reports (report_type, filename,
created_at) VALUES (?, ?, ?)",
                     (report_type, filename,
datetime.datetime.now().strftime("%Y-%m-%d %H:%M:%S")))
    return send_file(path, as_attachment=True, download_name=filename)
```

Після отримання результату необхідно записати його в базу даних, в нашому випадку в ролі бази даних виступає SQLite.

Крім запису у базу даних користувач має мати змогу завантажити файл, який був раніше згенерований та записаний у базу даних. За завантаження файлу відповідає функція `download_report`.

```
def download_report(filename):
    path = os.path.join(tempfile.gettempdir(), filename)
    if os.path.exists(path):
        return send_file(path, as_attachment=True, download_name=filename)
    return "Файл не знайдено", 404
```

Крім завантаження файлу потрібно тримати в голові, що файлу з будь-яких причин може не бути у базі даних. Тому якщо користувач захоче завантажити неіснуючий файл, потрібно видати помилку що файл не знайдено.

Для створення клієнтської частини веб-інтерфейсу було використано HTML, JavaScript та CSS. Основну увагу хотів би приділити саме JavaScript. Так як за допомогою нього було виконано з'єднання клієнської частини та серверної. Завдяки реалізації JavaScript-логіки у клієнтській частині веб-інтерфейсу, забезпечується зручна взаємодія з серверною частиною застосунку. Користувач може з легкістю генерувати нові звіти, переглядати попередні результати та завантажувати їх для подальшого аналізу. Функція `generateReport()` – головна

функція яка спрацьовує після натискання клавіші «Згенерувати та завантажити». Алгоритм роботи функції наступний:

1. Зчитує обраний тип аналізу з випадаючого списку
2. Відображає повідомлення про те що аналіз в роботі
3. Отримує файл аналізу та дозволяє його завантажити
4. Після завершення процесу повторно викликає функцію завантаження списку звітів `loadReports()`, щоб оновити інтерфейс.
5. У разі виникнення помилки, видає повідомлення у спеціальному блоці на сторінці

Для завантаження списку згенерованих звітів при оновлені сторінки використовується функція `loadReports()`. Принцип роботи функції простий, вона виконує GET-запит на маршрут `/api/reports`, який повертає список останніх 20 згенерованих репортів. У випадку якщо список порожній, на екран виведеться повідомлення «Список порожній»

3.3. Тестування програмного засобу

Тестування програмного засобу для автоматизації процесів операційного центру безпеки є важливим та необхідним кроком реалізації програмного засобу. Правильне тестування може виявити проблеми в роботі програмного засобу на перших етапах, що дозволить швидко їх виправити перед тим як використовувати програмний засіб в робочому середовищі. Першим етапом перевірки є тестування та перевірка інтерфейсу користувача. При перевірці та тестуванні інтерфейсу потрібно впевнитись що інтерфейс є інтуїтивно зрозумілий та не має візуальних багів. Після чого спробувати скористатись всім його функціоналом для виявлення проблем в роботі програмного засобу. На рисунку 3.3 зображено вигляд веб-інтерфейсу.

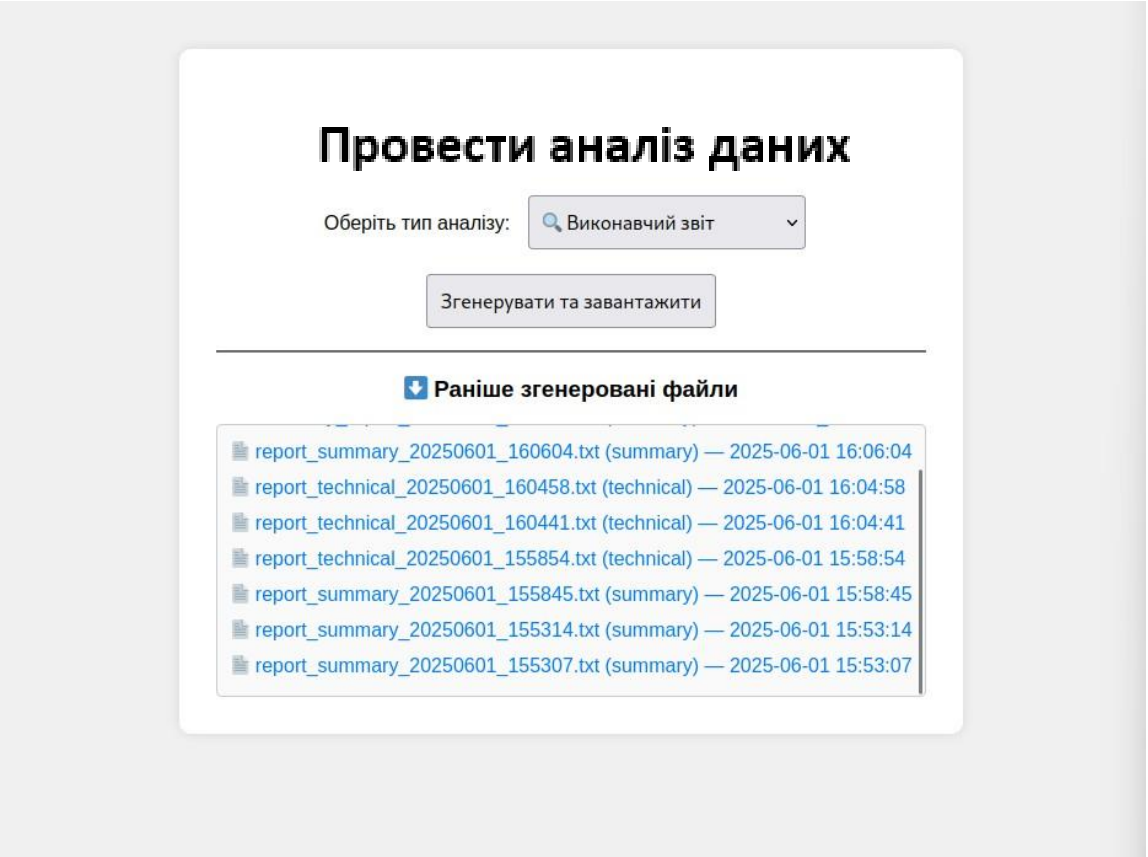


Рисунок 3.3 – Вигляд веб-інтерфейсу

Варто перевірити візуальну роботу веб інтерфейсу, а саме наскільки працюють випадаючі списки та повідомлення про створення аналізу. Крім перевірки анімації, важливо щоб те що бачить користувач відповідало дійсності. Тобто якщо користувач бачить надпис про створення звіту, звіт і вправду має створюватись, а не користувач бачить анімацію, після чого виявляється що система має внутрішні помилки. Тому, при тестуванні анаміції було також тестування дійсності показаного у веб-інтерфейсі. Також було протестовано, що кожен пункт випадаючого списку відповідає певному типу аналізу. Тестування було виконано шляхом завантаження файлів різних типів. Наприклад файл виконачого аналізу, та файл технічного аналізу. На рисунку 3.4 показано роботу випадаючих списків та повідомлення про створення звіту.

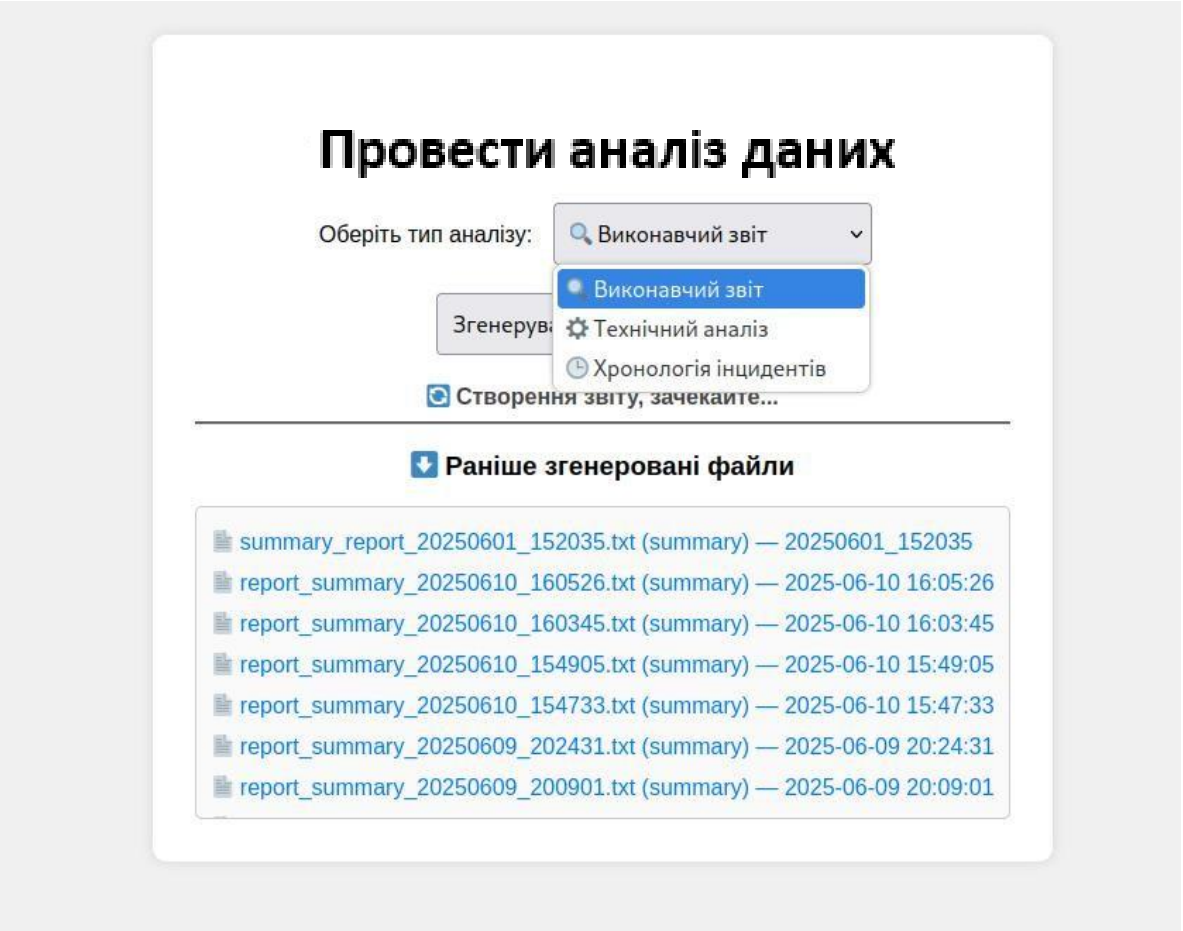


Рисунок 3.4 – Тестування роботи анімації, та випадаючого списку

Необхідно протестувати програму на випадок помилок різного роду, якщо виникла помилка з SIEM-системою, або штучним інтелектом, то користувач має зрозуміти що відбулось. Якщо такого не буде, він може подумати що проблема в браузері, мережевому з’єднанні або його комп’ютером. Протестуємо роботу програми у вигляді різних помилок, для прикладу замінемо API ключ для штучного інтелекту на неіснуючий (рис. 3.5).

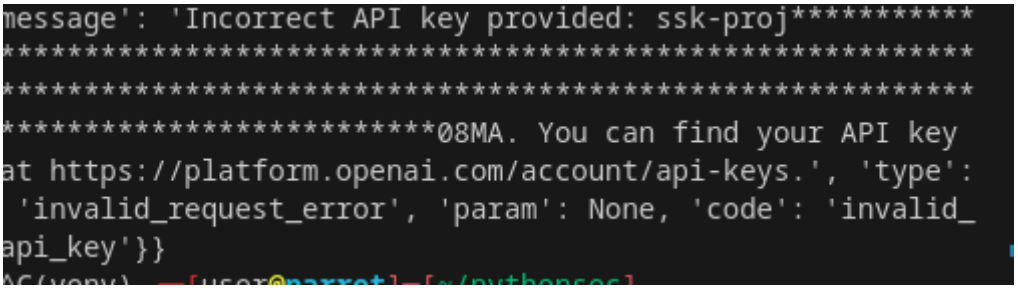


Рисунок 3.5 –Повідомлення після використання невірного API ключа

Протестуємо помилку на випадок проблеми з SIEM системою, заміним логін та пароль на несправжній, дивимось рисунок 3.6.

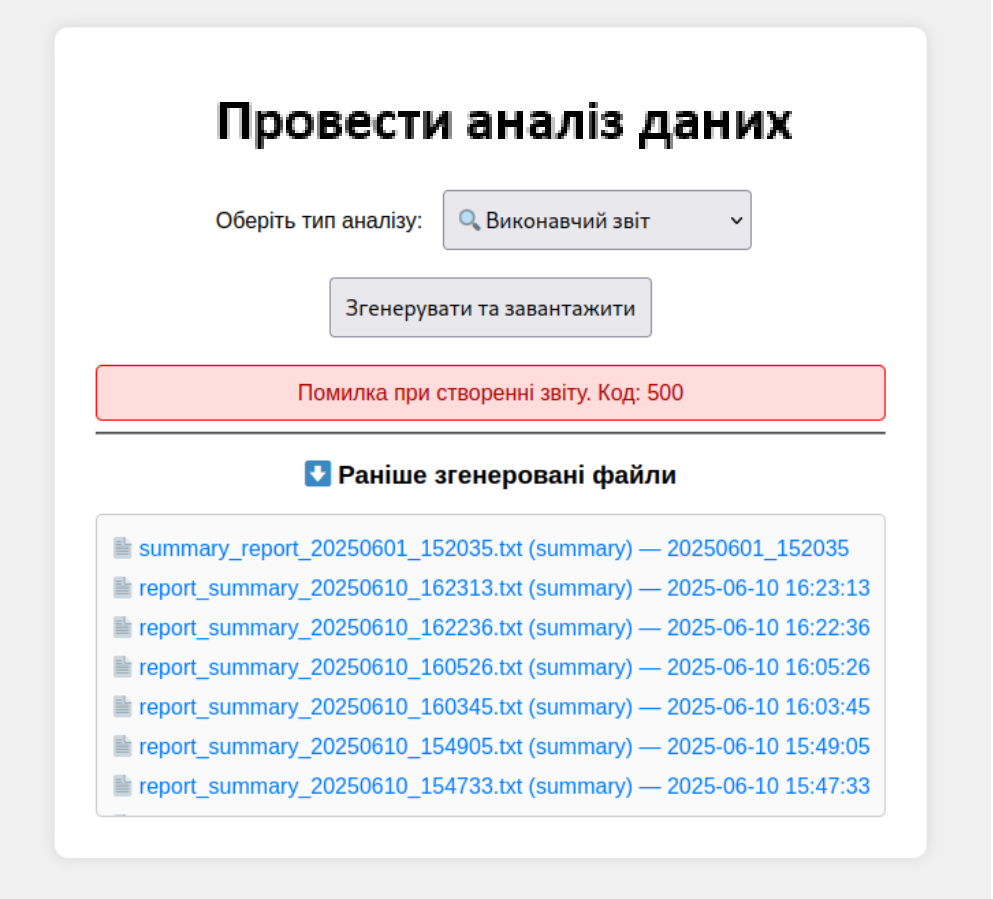


Рисунок 3.6 – Демонстрація проблеми з SIEM-системою

Протестуємо отримання файлів з бази даних, натиснемо на останній створений звіт, та перевіримо можливість його завантаження та вміст. Під час завантаження файлу, він має дефолтну назву, яку йому присвоїв програмний застосунок. Користувач має змогу змінити назву файлу, або залишити її як було. Назва файлу яку отримує користувач у випадку завантаження файлу без зміни його назви, хоча й має на перший погляд не дуже зрозумілий вигляд, але це легко дає змогу відрізняти однотипні файли між собою. Завантаження звіту з бази даних продемонстровано на рисунку 3.7.

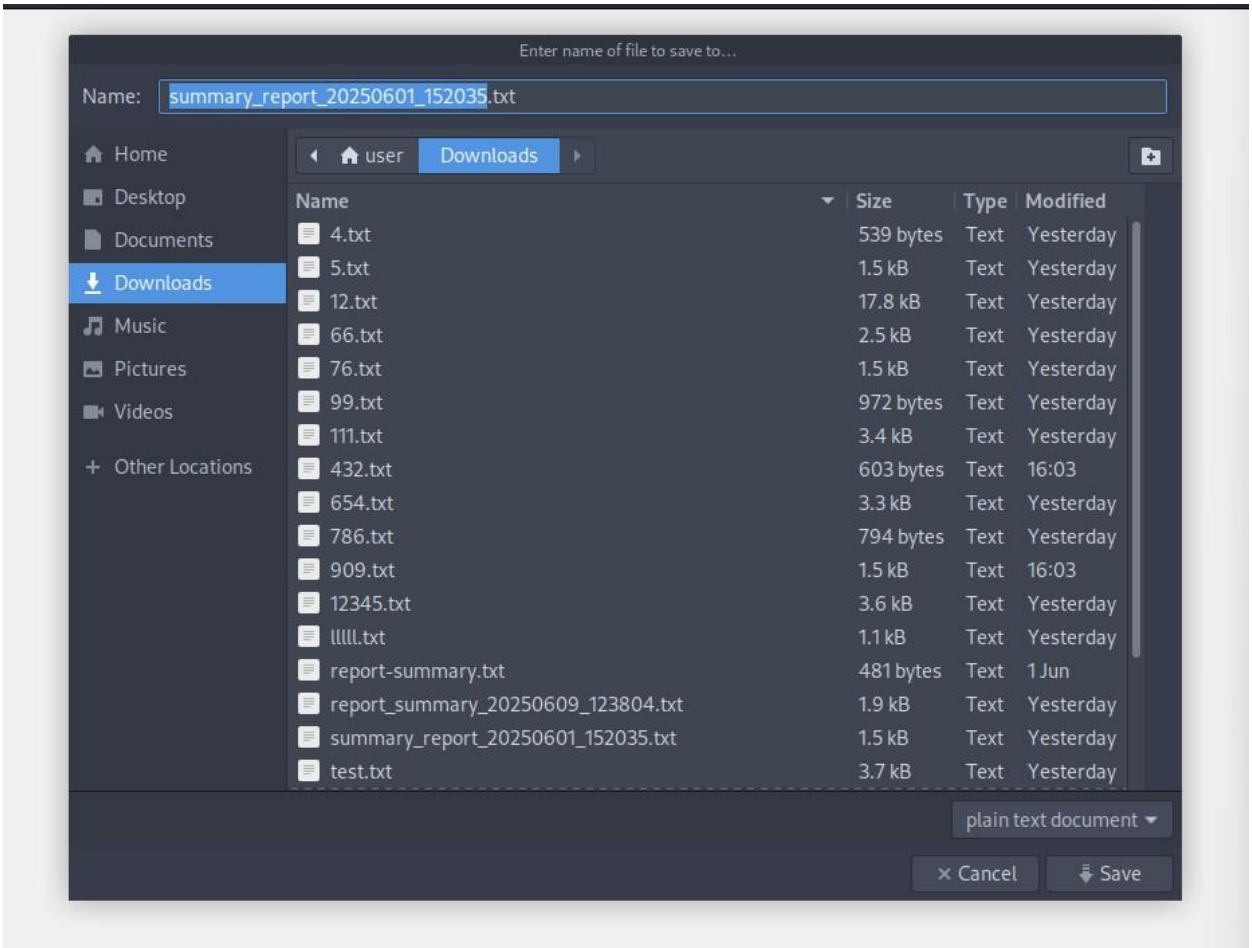


Рисунок 3.7 – Вікно завантаження файлу з бази даних

Другий етап тестування включає в себе перевірку змісту звітів, чи відповідають вони бажаній структурі, та чи мають правильний зміст. Якщо отриманий файл не відповідає бажаному вмісту, потрібно перевірити правильність написання промптів для великої мовної моделі. Також потрібно звернути увагу на температуру, у програмному застосунку використовується температура 0.1. Цього достатньо для генерації необхідних нам файлів з мінімальною кількістю неточностей та галюцинацій, на які здатен штучний інтелект. У випадку невдалого тестування потрібно виправити вище зазначані параметри та провести тестування до бажаного результату. Для тестування візьмемо виконавчий звіт, та завантажимо його, дивимось рисунок 3.8.

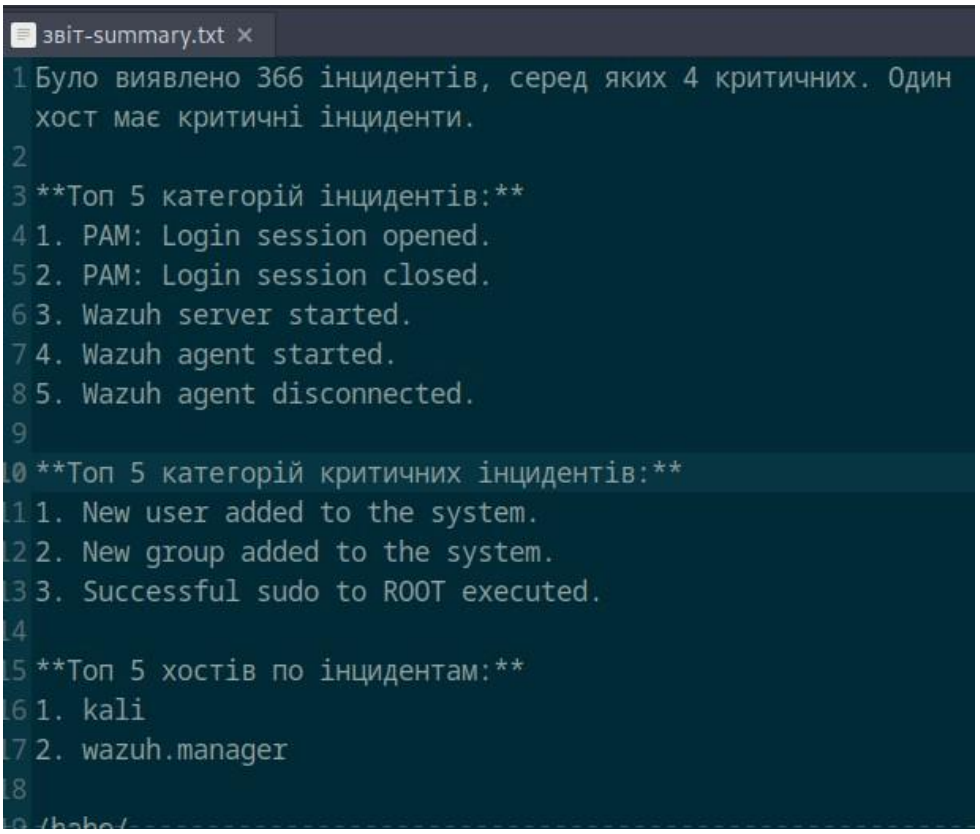


Рисунок 3.8 - Вміст та структура звіту відповідає бажаній

Як бачимо, звіт відповідає бажаній структурі яку ми задавали за допомогою користувацьких промптів, та зміст звіту правдивий, відповідає реальності.

Протестуємо час на аналіз даних та отримання результату від натискання на клавішу генерації звіту, до завантаження звіту на комп’ютер. Завантажимо кожен звіт по 10 разів та порахуємо середній час виконная аналізу. Перевірка відбувалась у різний час доби для точнішого результату, адже сервери OpenAI можуть бути завантажені по різному у різний час. Тому навіть при однакових вхідних та вихідних даних, швидкість може суттєво відрізнятись. Варто зазначити що на швидкість отримання результату впливає також швидкість інтернет з’єднання, тому при вимірах спочатку було проведення тестування швидкості інтернет з’єднання, та тільки після отримання задовільного результату, почались проводитись заміри швидкості роботи програмного застосунку.

Таблиця 3.9 – Заміри часу на виконання аналізу

	Виконавчий звіт	Технічний аналіз	Хронологія інцидентів
Середній час	11,3 секунди	9,4 секунди	10,3 секунди
Мінімальний час	6,8 секунд	6,2 секунди	8,2 секунди
Максимальний час	14,6 секунд	18,1 секунда	12,7 секунд

Для тестування точності програмного засобу, було встановлено агент на тестовий пристрій. Агент представляє собою програмне рішення від виробника Wzauh SIEM, яке встановлюється на системи Linux, Windows або MacOS, та передає системні події на SIEM-систему.

Агент було встановлено на пристрій з ОС Kali Linux, з використанням внутрішніх інструкцій по підключенню його до SIEM. Протестуємо підключення пристрою на рисунку 3.10.

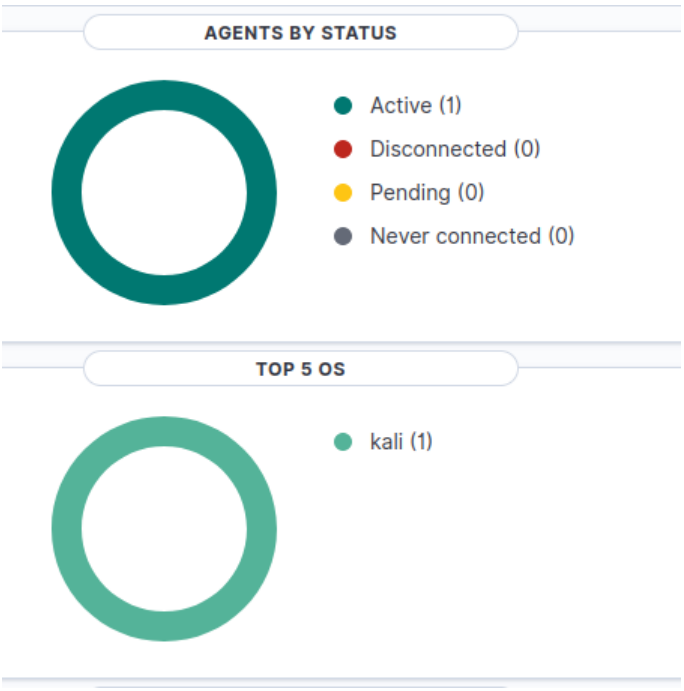


Рисунок 3.10 – Агент відображається у SIEM-системі після підключення

Для виконання тестування нам необхідно виконувати на кінцевому пристрої різні шкідливі дії, тоді пристрій передасть ці події на SIEM-систему, яка в свою чергу дозволить нам швидко проаналізувати дані подій за певний період. Проблема в тому, що мануально виконати сотні дій на кінцевому пристрої досить важко, тому потрібно зрозуміти як саме агент зчитує системні події. З’ясовано, що системні події записуються у наступні файли:

- "/var/log/apache2/access.log",
- "/var/log/apache2/error.log",
- "/var/log/dpkg.log",
- "/var/ossec/logs/active-responses.log"

Таким чином, якщо ми наповнимо ці файли системними подіями, агент буде зчитувати їх як реальні. Таким чином ми зможемо наповнити SIEM-систему сотнями логів, та протестувати точність аналізу даних нашого програмного застосунку.

Для наповненя файлів системними подіями, був розроблений скрипт, який потрібно запускати на кінцевому девайсі. Для більш точного тестування потрібно враховувати різні сценарії. Наприклад, коли в нас всі події не мають високого рівня критичності, або багато подій мають високий рівень загроз. Таким чином ми зможемо перевірити програмний засіб на точності чи галюцинації. Крім того ми все ще можемо мануально виконувати дії, які будуть розцінені SIEM-системою як шкідливі, щоб перевірити точність після додавання одної, або декількох подій. Перевіримо кількість загроз до наповненя файлів системних подій, та виконання скрипту. Для отримання правильної роботи скрипту тестування, спочатку провівся аналіз на невеликій кількості системних подій, після чого відбулось повноцінне наповнення системних файлів системними подіями, для тестування роботи програмного застосунку. На рисунку 3.11 показано кількість загроз до виконання скрипту.

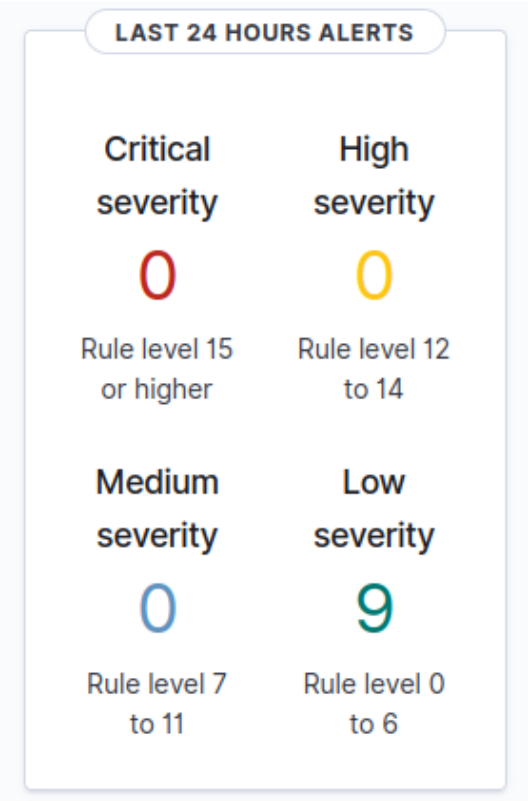


Рисунок 3.11 – Кількість подій до запуску скрипту

Після запуску скрипту з функцією генерування подій низької шкідливості, перевіримо їх кількість у SIEM-системі. Перейдемо на головну сторінку SIEM-системи та оновимо її, тепер можемо бачити що загальна кількість подій зросла з 9 до 276. Таким чином скрипт згенерував 267 системні події. Крім того всі системні події відносяться до низького класу, тобто не несуть серйозної загрози. У цьому випадку згенерований файл має містити у собі запис про те, що всі події були низького класу критичності, та не мають у собі критичних подій, які можуть викоикати проблеми у системі. Крім того, варто зазначити що цей вид тестування зможе виявити проблеми у розумінні системних та користувацьких промптів [19]. Так як ми явно вказали спосіб думок, та використовували інші техніки промпт інженерії, які дозволяють нам уникнути проблем при аналізі даних із SIEM-системи. Рисунок 3.12 показує кількість подій після запуску та виконання скрипту.

Рисунок 3.12 – Кількість інцидентів після виконання скрипта.

Зайдемо на веб-інтерфейс програмного засобу для аналізу даних та згенеруємо простий виконавчий звіт по подіям. Завантажимо звіт та перевіримо його працездатність. Звіт завантажується у текстовому форматі, та користувач може легко змінити назву файлу та директорію у яку буде завантажуватись звіт. Цього було досягнено завдяки використанню браузера та веб інтерфейсу. Нам не потрібно додавати це у функціонал програми вручну, що значно полегшує розробку та доповнення програмного застосунку. У завантаженому файлі можемо бачити, що система виявила всі 276 інцидентів, та правильно їх згрупувала. Серед інцидентів не було кричних, тому у звіті чітко сказано що жоден інцидент не є критичним. Крім того зазначено що всі інциденти стосуються одного хоста, так як у нас лише один тестовий хост. На рисунку 3.13 показано результат аналізу.

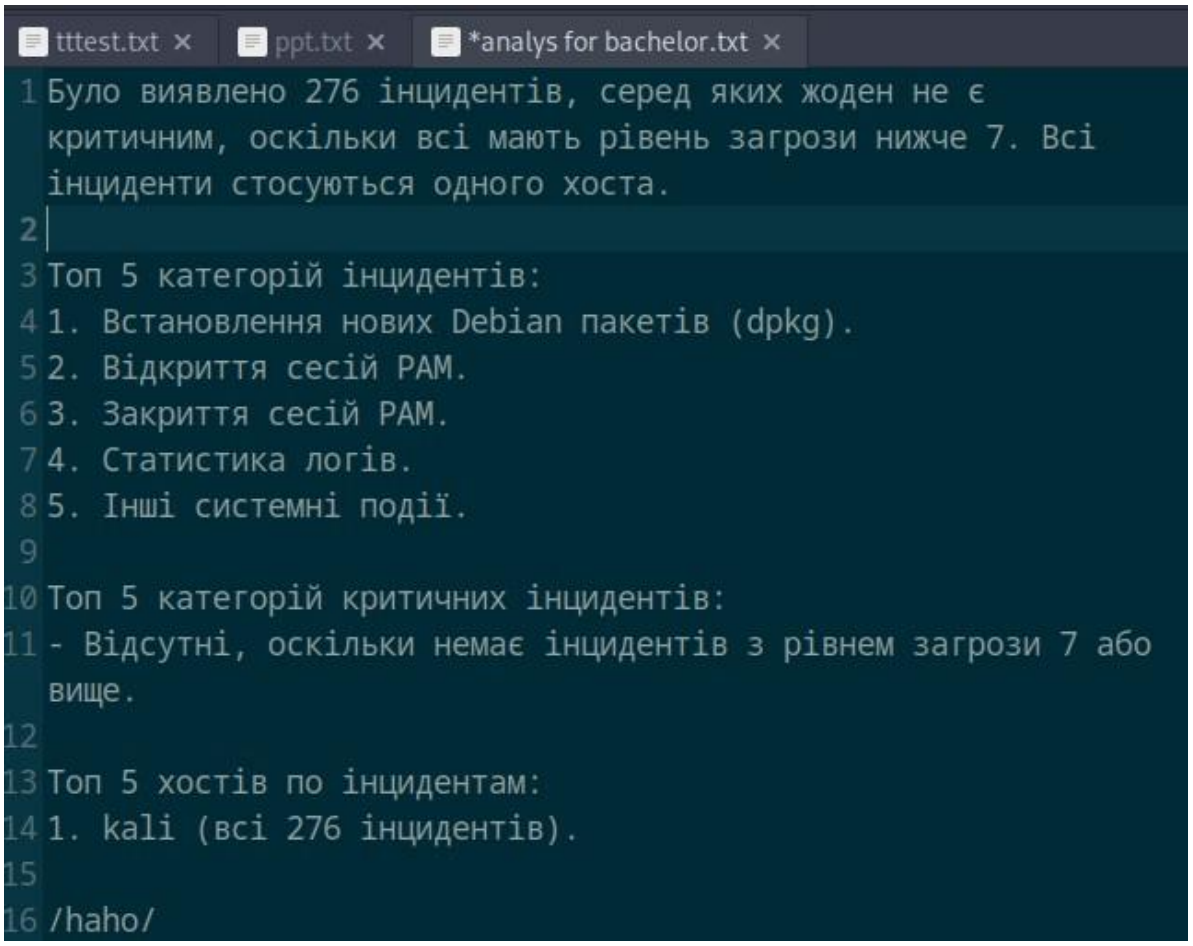


Рисунок 3.13 – Результат виконавчого звіту

Додамо декілька інцидентів які можна вважати критичними, зробимо це для тестування системи аналітики, чи виявить вона критичні інциденти. Додамо критичні інциденти наступними командами на кінцевому пристрої:

```
sudo useradd eviladmin
sudo usermod -aG sudo eviladmin
```

Таким чином ми заставимо агент повідомити SIEM-систему про 2 критичних інцидента. На рисунку 3.14 можна побачити виведення інцидентів в SIEM-системі. Після цього згенеруємо повторний виконавчий звіт. Після чого впевнемося у коректних результатах тестування. На рисунку 3.15 можна побачити звіт як на рисунку 3.13, з додаванням двох критичних інцидентів.

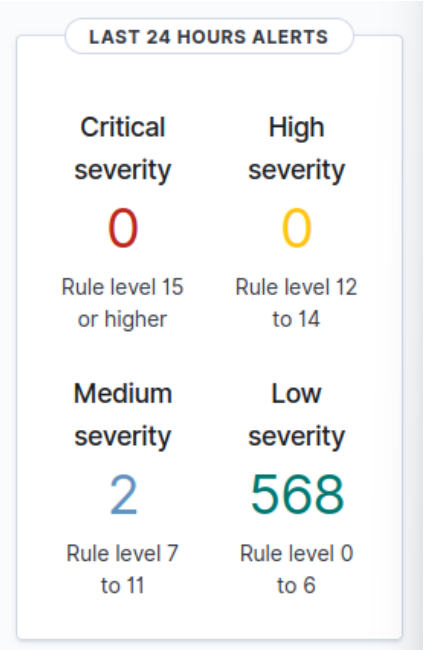


Рисунок 3.14 - Кількість загроз у SIEM-системі

```
Було виявлено 568 інцидентів, серед яких 2 критичних. Один
хост має критичні інциденти.

Топ 5 категорій інцидентів:
1. PAM: Login session opened.
2. PAM: Login session closed.
3. Successful sudo to ROOT executed.
4. New dpkg (Debian Package) requested to install.
5. New user added to the system.

Топ 5 категорій критичних інцидентів:
1. New user added to the system.
2. New group added to the system.

Топ 5 хостів по інцидентам:
1. kali
2. wazuh.manager

/haho/
```

Рисунок 3.15 – Звіт після додавання двох критичних інцидентів

3.4. Порівняння робочих задач до та після автоматизації за допомогою програмного засобу

До впровадження автоматизації процесу генерації звітів у сфері кібербезпеки, значна частина аналітичної роботи виконувалась вручну. Це включало підготовку технічних звітів, аналіз логів та інцидентів, формулювання висновків і створення виконавчих резюме для керівництва. Такі завдання, як правило, покладалися на аналітиків безпеки, які мусили взаємодіяти з SIEM-системами (наприклад, Wazuh, Splunk тощо), вручну переглядати та сортувати події, оцінювати рівень критичності, категоризувати інциденти та оформлювати результати у зрозумілий для управлінської ланки вигляд.

Оскільки кожен звіт потребував ретельного аналізу великої кількості даних, процес займав значну кількість часу - в залежності від типу даних від 10 хвилин, до години, залежно від обсягу інформації. Крім того, при створенні технічного звіту існував ризик помилок або упущення критичних деталей через людський фактор або втому. Ще однією проблемою була відсутність єдиної структури звітів - різні спеціалісти використовували власні підходи, шаблони та формулювання. Це ускладнювало стандартизацію звітності та її подальше використання в аналітичних цілях.

З впровадженням розробленого програмного засобу процес створення звітів став значно ефективнішим і структурованішим. Інструмент автоматично здійснює запити до SIEM-системи та отримує останні події з логів безпеки. Далі ці дані передаються великій мовній моделі, яка формує звіт відповідно до обраного типу: виконавчий підсумок, технічний аналіз або хронологія подій. Цей процес займає лічені хвилини, і звіт одразу готовий до використання.

Однією з головних переваг автоматизації стало скорочення часу, необхідного для створення звіту. Аналітики більше не витрачають години на пошук, класифікацію та структурування інформації. Замість цього вони можуть

зосередитися на прийнятті рішень, подальшому аналізі або розробці заходів реагування. Це значно підвищує загальну продуктивність команди та дозволяє ефективніше використовувати людські ресурси.

Автоматизована система також покращила якість комунікації між технічною та управлінською командами. Завдяки наявності виконавчих звітів, написаних простою мовою, керівництво може швидко ознайомитися з поточним станом безпеки інфраструктури, оцінити ризики та приймати обґрунтовані управлінські рішення. Відсутність необхідності запитувати звіти вручну знижує навантаження на команду безпеки та зменшує кількість непотрібної бюрократії.

Ще одним важливим аспектом є підвищення прозорості та стандартизації. Автоматично згенеровані звіти мають уніфіковану структуру, що полегшує подальше зберігання, аналіз і порівняння. Це відкриває нові можливості для аналітики, наприклад, оцінку ефективності заходів безпеки у динаміці або виявлення повторюваних типів атак.

Окрім технічної зручності, автоматизація також дозволила команді зосередитися на більш пріоритетних завданнях. Рутинна звітність, яка раніше займала значну частину часу, тепер виконується системою без втручання людини. Це зменшило рівень стресу в команді, покращило моральний клімат і підвищило якість загального моніторингу.

Значну роль у змінах відіграє й точність аналізу. Велика мовна модель, яка виконує функцію аналізатора даних, здатна швидко обробляти великі обсяги інформації, виявляючи патерни, які могли б залишитися поза увагою аналітика. Крім того, модель не втомлюється, не відволікається і завжди дотримується заданого шаблону — це мінімізує ймовірність помилок.

Також важливо зазначити, що автоматизований підхід є масштабованим. У великих організаціях або в умовах інтенсивного зростання кількості подій, система з ручною генерацією звітів швидко стає вузьким місцем. Завдяки автоматизації, навантаження може зростати без втрати якості або продуктивності, що критично для сучасних інформаційно-насичених середовищ.

Загалом, перехід від ручного до автоматизованого підходу можна описати як еволюцію в напрямку ефективності, стандартизації та прозорості. Якщо раніше звіт був лише формальністю, то тепер він перетворився на дійсно корисний інструмент прийняття рішень, доступний у режимі реального часу. Саме це дозволяє говорити про суттєве підвищення рівня інформаційної безпеки, організованості робочих процесів та зрілості операційного центру безпеки загалом.

ВИСНОВКИ

Метою даної роботи було створення програмного засобу для автоматизації процесів першого рівня операційного центру безпеки з використанням штучного інтелекту. Основною задачею стало зменшення рутинного навантаження на фахівців, автоматичне формування звітів на основі даних з SIEM-системи та надання простого інтерфейсу для керування процесом аналізу.

У процесі реалізації були досягнуті всі поставлені задачі: розроблено веб-застосунок з інтуїтивно зрозумілим інтерфейсом, створено модуль збору та обробки логів з SIEM-системи Wazuh, налагоджено інтеграцію з великою мовною моделлю GPT-4o через API, реалізовано збереження звітів у базу даних SQLite та можливість їх повторного завантаження.

Використання великих мовних моделей у сфері автоматизації операційного центру безпеки показало свою ефективність та перспективність. Мовна модель продемонструвала здатність обробляти великі обсяги логів, структуровано подавати інформацію, робити аналітичні висновки та формувати звіти у форматі, зрозумілому як технічним спеціалістам, так і керівництву.

Автоматизація за допомогою LLM не лише підвищує ефективність реагування на інциденти, а й дозволяє аналітикам зосередитись на критично важливих завданнях. Програмний засіб, створений у рамках цієї роботи, може стати основою для подальшої розробки повноцінного інструменту аналітики першого рівня SOC або бути інтегрованим у більші системи інформаційної безпеки.

Таким чином, результати роботи доводять доцільність і практичну користь впровадження штучного інтелекту в процеси кібербезпеки.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. QBE. Connected Business: digital dependency fuelling risk. URL: https://qbeeurope.com/news-and-events/press-releases/annual-global-cyber-attacks-double-from-2020-to-2024-qbe/ (дата звернення: 21.05.2025).
2. 2Wired. Ransomware Payments Hit a Record $1.1 Billion in 2023. URL: https://www.wired.com/story/ransomware-payments-2023-breaks-record/ (дата звернення: 21.05.2025).
3. AInvest. Silent Storm: Cybersecurity Vulnerabilities in Energy Infrastructure. URL: https://www.ainvest.com/news/silent-storm-cybersecurity-vulnerabilities-energy-infrastructure-wake-call-investors-2505/ (дата звернення: 21.05.2025).
4. Forbes. DeepSeek Data Leak Exposes 1,000,000 Sensitive Records. URL: https://www.forbes.com/sites/larsdaniel/2025/02/01/deepseek-data-leak-exposes--1000000-sensitive-records/ (дата звернення: 21.05.2025).
5. Proofpoint. 2025 State of the Phish Report. URL: https://www.proofpoint.com/us/resources/threat-reports/state-of-phish (дата звернення: 21.05.2025).
6. IBM Security X-Force. Threat Intelligence Index 2024. URL: https://www.ibm.com/reports/threat-intelligence (дата звернення: 21.05.2025).
7. (ISC)². Cybersecurity Workforce Study, 2024. URL: https://www.isc2.org/Research/Cybersecurity-Workforce-Study (дата звернення: 21.05.2025).
8. SOC Analyst Burnout: Essential Steps to Minimize It with AI https://radiantsecurity.ai/learn/soc-analyst-burnout/ (дата звернення: 21.05.2025)
9. LLM: що це таке і які відкриває можливості. Anywhere Club. URL: https://aw.club/global/uk/blog/what-are-llms-and-what-opportunities-do-they-offer (дата звернення: 21.05.2025).
10. OpenAI Assistants. URL: https://platform.openai.com/docs/assistants/how-itworks/objects/ (дата звернення: 21.05.2025).

11. OpenAI API. Jun 11, 2020. URL: https://openai.com/index/openai-api/ (дата звернення: 21.05.2025).
12. OpenAI GPTs. Nov 6, 2023. URL: https://openai.com/index/introducing-gpts/ (дата звернення: 21.05.2025)
13. SOC -everything you need to know https://www.iver.com/en/press-articles/soc---everything-you-need-to-know-about-security-operations-centers/ (дата звернення: 21.05.2025)
14. SOC https://www.splunk.com/en_us/blog/learn/soc-security-operation-center.html (дата звернення: 21.05.2025).
15. Evolution of SOC https://thehackernews.com/2025/02/soc-30-evolution-of-soc-and-how-ai-is.html (дата звернення: 21.05.2025).
16. What is SOC https://builtin.com/articles/soc-security-operations-center (дата звернення: 21.05.2025).
17. Що таке python і навіщо він потрібен? URL: https://hyperhost.ua/info/uk/shhotake-python-i-navishho-vinpotriben?gad_source=1&gclid=Cj0KCQjw0ruyBhDuARIsANSZ3wqPnmijVZTeMAOBStJ40_DZtX8zIX4QjYZyAGQ8dx0nS3D5-hrdc4aAvsgEALw_wcB (дата звернення: 21.05.2025).
18. Wazuh documentation https://documentation.wazuh.com/current/index.html (дата звернення: 21.05.2025)
19. Wazuh capabilities data collection https://documentation.wazuh.com/current/user-manual/capabilities/log-data-collection/how-it-works.html (дата звернення: 21.05.2025)
20. Wazuh capabiliries user manual https://documentation.wazuh.com/current/user-manual/capabilities/index.html (дата звернення: 21.05.2025)

ДОДАТКИ

64

Додаток А

ПРОТОКОЛ ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ

Назва роботи: Засіб для автоматизації процесів операційного центру безпеки на основі штучного інтелекту

Автор роботи: Антонюк Олексій Максимович

Тип роботи: бакалаврська кваліфікаційна робота

Підрозділ кафедра захисту інформації ФІТКІ, група 1 БКС-21 б

Коефіцієнт подібності текстових запозичень, виявлених у роботі системою StrikePlagiarism **0,26** %

Висновок щодо перевірки кваліфікаційної роботи (відмітити потрібне)

√ Запозичення, виявлені у роботі, є законними і не містять ознак плагіату, фабрикації, фальсифікації. Роботу прийняти до захисту

☐ У роботі не виявлено ознак плагіату, фабрикації, фальсифікації, але надмірна кількість текстових запозичень та/або наявність типових розрахунків не дозволяють прийняти рішення про оригінальність та самостійність її виконання. Роботу направити на доопрацювання.

☐ У роботі виявлено ознаки плагіату та/або текстових маніпуляцій як спроб укриття плагіату, фабрикації, фальсифікації, що суперечить вимогам законодавства та нормам академічної доброчесності. Робота до захисту не приймається.

Експертна комісія:

Завідувач кафедри ЗІ д. т. н., проф. _____ Володимир ЛУЖЕЦЬКИЙ

Гарант освітньої програми «Кібербезпека критичних систем» к. т. н., доц. _____ Юрій БАРИШЕВ

Особа, відповідальна за перевірку _____ Валентина КАПЛУН

З висновком експертної комісії ознайомлений(-на)

Керівник _____ Леонід КУПЕРШТЕЙН

Здобувач _____ Олексій АНТОНЮК

Додаток Б

Текст програми Додаток

```
from flask import Flask, request, send_file, jsonify
from openai import OpenAI
import os
import tempfile
from dotenv import load_dotenv
import sqlite3
import datetime
import requests
import json

# Load secrets from .env
load_dotenv()
client = OpenAI(api_key=os.getenv("OPENAI_API_KEY"))
WAZUH_USER = os.getenv("WAZUH_USER")
WAZUH_PASS = os.getenv("WAZUH_PASS")
WAZUH_URL = "https://localhost:9200/wazuh-alerts-*/_search?pretty"

app = Flask(__name__)
DB_NAME = "reports.db"

PROMPTS = {
    "summary": (
        "Створи короткий звіт по системним подіям. Звіт має наступну
структуру: 2-4 речення висновку простими словамию Наприклад, було виявлено
22 інцидента, серед яких 5 критичних, 3 хоста мають критичні інциденти.
далі наведи списком: Топ 5 категорій інцидентів, топ 5 категорій критичних
інцидентів. Топ 5 хостів по інцидентам.далі наведи списком: не потрібно
давати рекомендацій, просто аналіз даних.  в кінці напиши /haho/ для того
щоб я бачив що ти зрозумів промпти"
    ),
    "technical": (
        "Проаналізуй події технічно. Для кожного інциденту: опис, тип
атаки, IP, користувач, рівень загрози.   "
        ""
```

```
    ),
    "timeline": (
        "Створи хронологічний звіт інцидентів. Покажи час, опис події,
зміну поведінки атакуючих. "
        "Впорядкуй події за часом."
    )
}

with sqlite3.connect(DB_NAME) as conn:
    conn.execute("""
        CREATE TABLE IF NOT EXISTS reports (
            id INTEGER PRIMARY KEY AUTOINCREMENT,
            report_type TEXT,
            filename TEXT,
            created_at TEXT
        )
    """)

HTML_PAGE = '''
<!DOCTYPE html>
<html lang="uk">
<head>
  <meta charset="UTF-8" />
  <title>Генератор звітів Wazuh</title>
  <style>
    body { font-family: Arial, sans-serif; background: #f0f0f0; text-
align: center; padding: 50px; }
    .container { background: #fff; padding: 30px; border-radius: 10px;
display: inline-block; box-shadow: 0 0 10px rgba(0,0,0,0.1); }
    select, button { padding: 10px; margin: 10px; font-size: 16px; }
    #loading { display: none; font-weight: bold; margin-top: 10px; color:
#444; }
    #errorBox { display: none; margin-top: 10px; padding: 10px;
background: #ffdddd; border: 1px solid red; border-radius: 5px; color:
#a00; }
    #reportsList {
      max-height: 200px;
      overflow-y: auto;
      text-align: left;
      margin-top: 10px;
      border: 1px solid #ccc;
      padding: 10px;
      background: #fafafa;
      border-radius: 5px;
    }
    #reportsList a {
      display: block;
      padding: 5px 0;
      color: #007BFF;
```

```
      text-decoration: none;
    }
    #reportsList a:hover {
      text-decoration: underline;
    }
  </style>
</head>
<body>
  <div class="container">
    <h1>Згенерувати аналіз даних</h1>

    <label for="reportType">Оберіть тип аналізу:</label>
    <select id="reportType">
      <option value="summary"> Виконавчий звіт</option>
      <option value="technical"> Технічний аналіз</option>
      <option value="timeline"> Хронологія інцидентів</option>
    </select>

    <br />
    <button onclick="generateReport()">Згенерувати та завантажити</button>
    <div id="loading">  Створення звіту, зачекайте...</div>
    <div id="errorBox"></div>

    <hr />
    <h3>    Раніше згенеровані файли</h3>
    
    <div id="reportsList">Завантаження...</div>
  </div>

  <script>
    async function generateReport() {
      const type = document.getElementById("reportType").value;
      const loading = document.getElementById("loading");
      const errorBox = document.getElementById("errorBox");

      loading.style.display = "block";
      errorBox.style.display = "none";
      errorBox.textContent = "";

      try {
        const res = await fetch("/api/report", {
          method: "POST",
          headers: { "Content-Type": "application/json" },
          body: JSON.stringify({ report_type: type })
        });

        if (!res.ok) throw new Error("Помилка при створенні звіту. Код: "
+ res.status);
```

```
        const blob = await res.blob();
        const url = window.URL.createObjectURL(blob);
        const a = document.createElement("a");
        a.href = url;
        a.download = `звіт-${type}.txt`;
        document.body.appendChild(a);
        a.click();
        document.body.removeChild(a);

        await loadReports();
      } catch (e) {
        errorBox.style.display = "block";
        errorBox.textContent = e.message;
      } finally {
        loading.style.display = "none";
      }
    }

    async function loadReports() {
      const res = await fetch("/api/reports");
      const data = await res.json();
      const list = document.getElementById("reportsList");

      if (data.length === 0) {
        list.innerHTML = "Немає звітів.";
        return;
      }

      list.innerHTML = "";
      data.forEach(r => {
        const link = document.createElement("a");
        link.href = `/download/${r.filename}`;
        link.textContent = `  ${r.filename} (${r.report_type}) —
${r.created_at}`;
        list.appendChild(link);
      });
    }

    loadReports();
  </script>
</body>
</html>
'''

@app.route("/")
def index():
    return HTML_PAGE

@app.route("/api/report", methods=["POST"])
```

```
def generate_report():
    data = request.get_json()
    report_type = data.get("report_type", "summary")
    prompt = PROMPTS.get(report_type, PROMPTS["summary"])

    wazuh_response = requests.get(
        WAZUH_URL,
        auth=(WAZUH_USER, WAZUH_PASS),
        headers={"Content-Type": "application/json"},
        json={
            "size": 50,
            "sort": [
                {"@timestamp": {"order": "desc"}}
            ],
            "query": {
                "range": {
                    "@timestamp": {
                        "gte": "now-24h",
                        "lte": "now"
                    }
                }
            }
        },
        verify=False
    )

    if wazuh_response.status_code != 200:
        return jsonify({"error": "Не вдалося отримати дані з Wazuh"}), 500

    alerts_data = wazuh_response.text

    messages = [
        {"role": "system", "content": (
            "Ти — аналітик кібербезпеки SOC рівня L1. "
            "Твоя задача — проаналізувати логи з SIEM-системи Wazuh, які
надходять у форматі JSON. "
            "Вхідні дані містять такі поля: @timestamp (час події),
rule.description (опис інциденту), "
            "rule.level (рівень загрози від 0 до 15), agent.name (назва
хоста), full_log (повне повідомлення), "
            "location (джерело логів), mitre (MITRE ATT&CK), data.dstuser,
data.srcip тощо. "
            "Якщо level >= 7 — вважай це критичною подією. "
            "На основі логів згенеруй відповідь у залежності від типу
звіту. "
            "Пиши українською мовою. Структурованість, узгодженість і
логіка — обов’язкові. "
            "Відповідь має бути у вигляді аналітичного тексту, зрозумілого
для людини. "
```

```
            "Уникай шаблонних фраз, поясню контекст."
        )},
        {"role": "user", "content": f"{prompt}\n\n{alerts_data}"}
    ]

    completion = client.chat.completions.create(
        model="gpt-4o",
        messages=messages,
        temperature = 0.1
    )

    result_text = completion.choices[0].message.content
    filename =
f"report_{report_type}_{datetime.datetime.now().strftime('%Y%m%d_%H%M%S')}
.txt"
    path = os.path.join(tempfile.gettempdir(), filename)

    with open(path, "w", encoding="utf-8") as f:
        f.write(result_text)

    with sqlite3.connect(DB_NAME) as conn:
        conn.execute("INSERT INTO reports (report_type, filename,
created_at) VALUES (?, ?, ?)",
                     (report_type, filename,
datetime.datetime.now().strftime("%Y-%m-%d %H:%M:%S")))

    return send_file(path, as_attachment=True, download_name=filename)

@app.route("/api/reports")
def get_reports():
    with sqlite3.connect(DB_NAME) as conn:
        cursor = conn.execute("SELECT report_type, filename, created_at
FROM reports ORDER BY created_at DESC LIMIT 20")
        reports = [dict(zip(["report_type", "filename", "created_at"],
row)) for row in cursor.fetchall()]
    return jsonify(reports)

@app.route("/download/<filename>")
def download_report(filename):
    path = os.path.join(tempfile.gettempdir(), filename)
    if os.path.exists(path):
        return send_file(path, as_attachment=True, download_name=filename)
    return "Файл не знайдено", 404

if __name__== "__main__":
    app.run(debug=True)
```

Додаток В

ІЛЮСТРАТИВНА ЧАСТИНА

ЗАСІБ ДЛЯ АВТОМАТИЗАЦІЇ ПРОЦЕСІВ ОПЕРАЦІЙНОГО ЦЕНТРУ БЕЗПЕКИ НА ОСНОВІ ШТУЧНОГО ІНТЕЛЕКТУ

Виконав: студент 4 курсу групи 1БКС-21б
спеціальності 125 Кібербезпека
_____ Олексій АНТОНЮК
16 червня 2025 р.

Керівник: к. т. н., доц., доцент каф. ЗІ
_____ Леонід КУПЕРШТЕЙН
16 червня 2025 р.

АРХІТЕКТУРА ЗАСОБУ АВТОМАТИЗАЦІЇ ПРОЦЕСІВ ОПЕРАЦІЙНОГО ЦЕНТРУ БЕЗПЕКИ НА ОСНОВІ ШІ

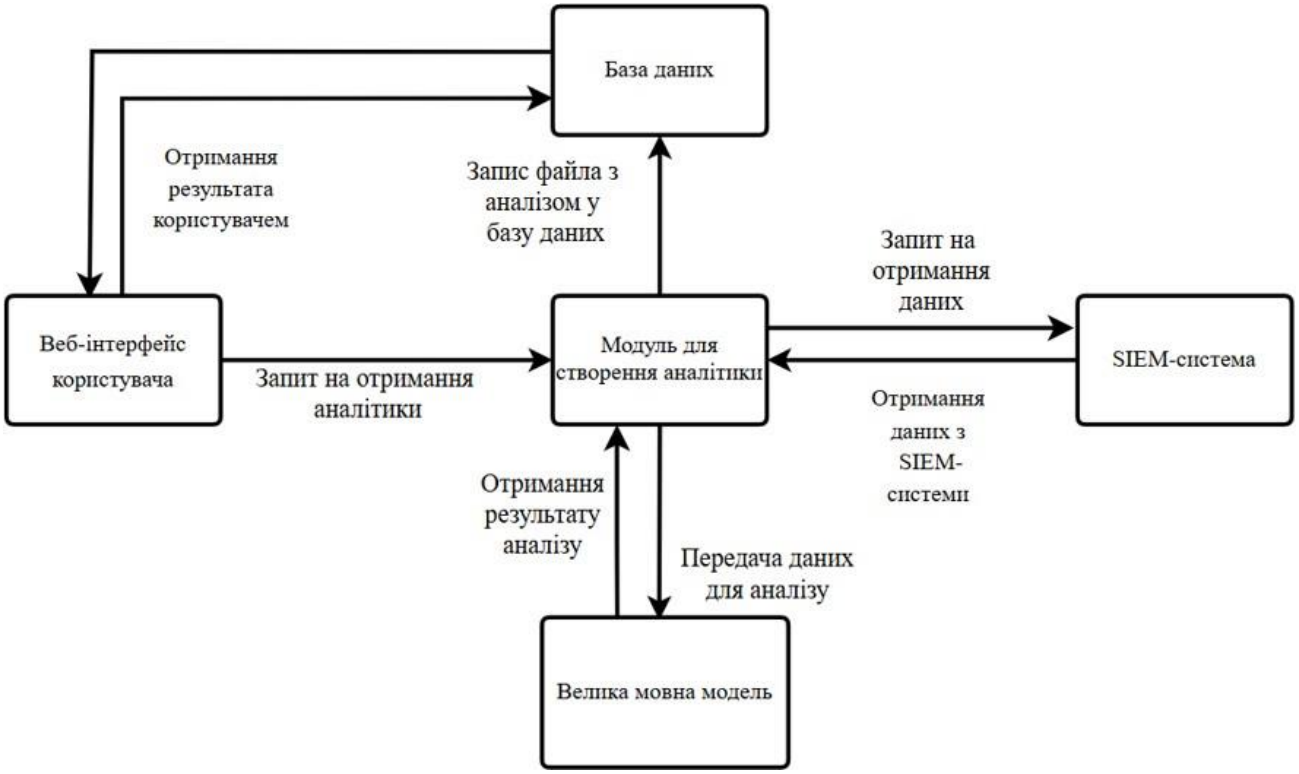
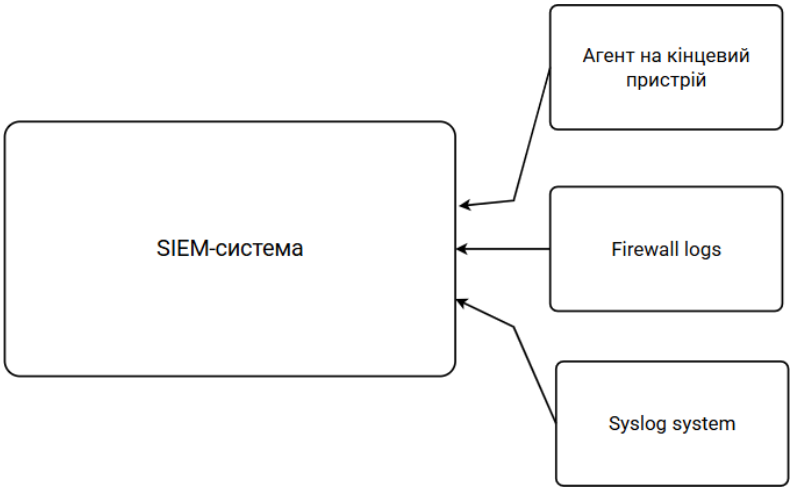
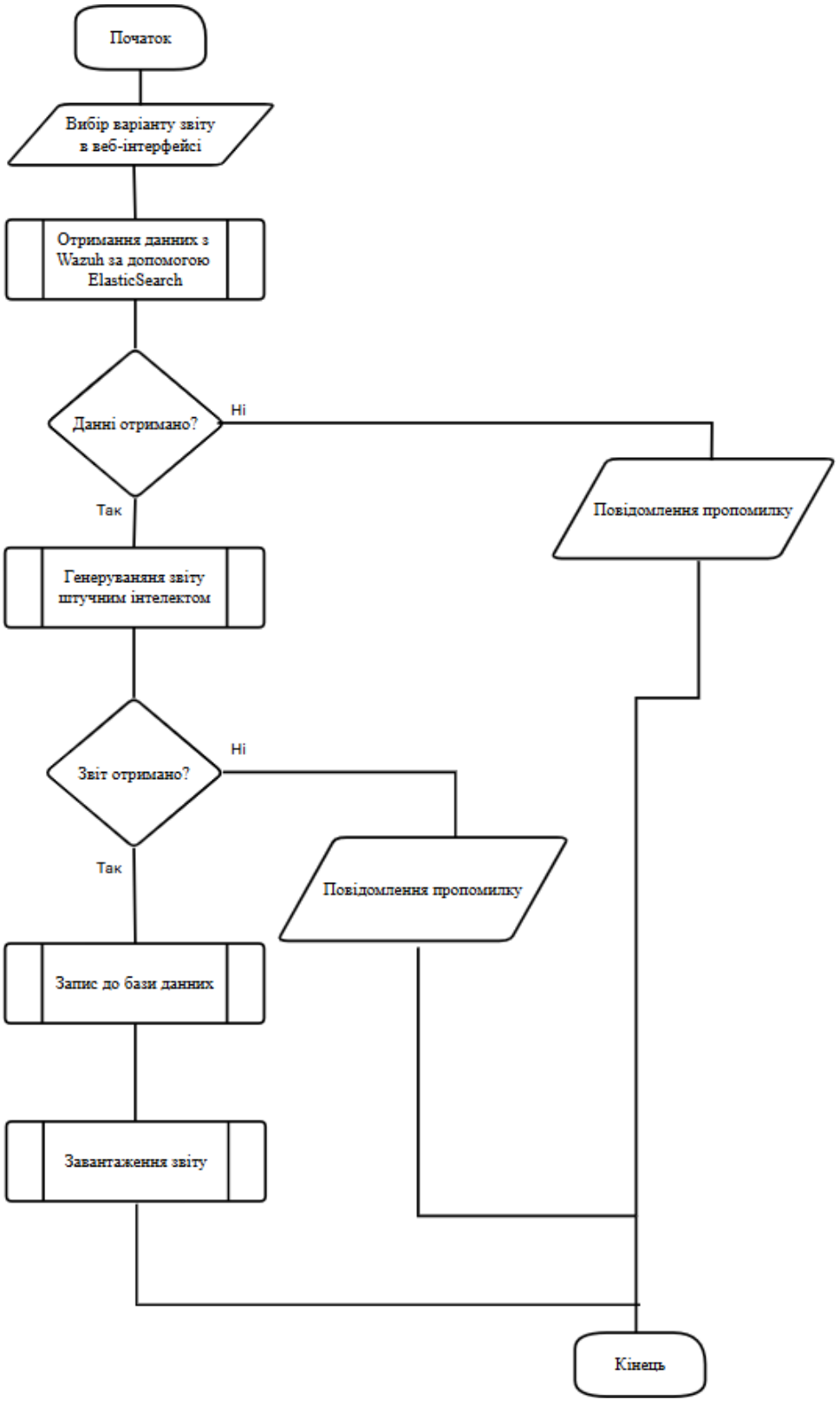


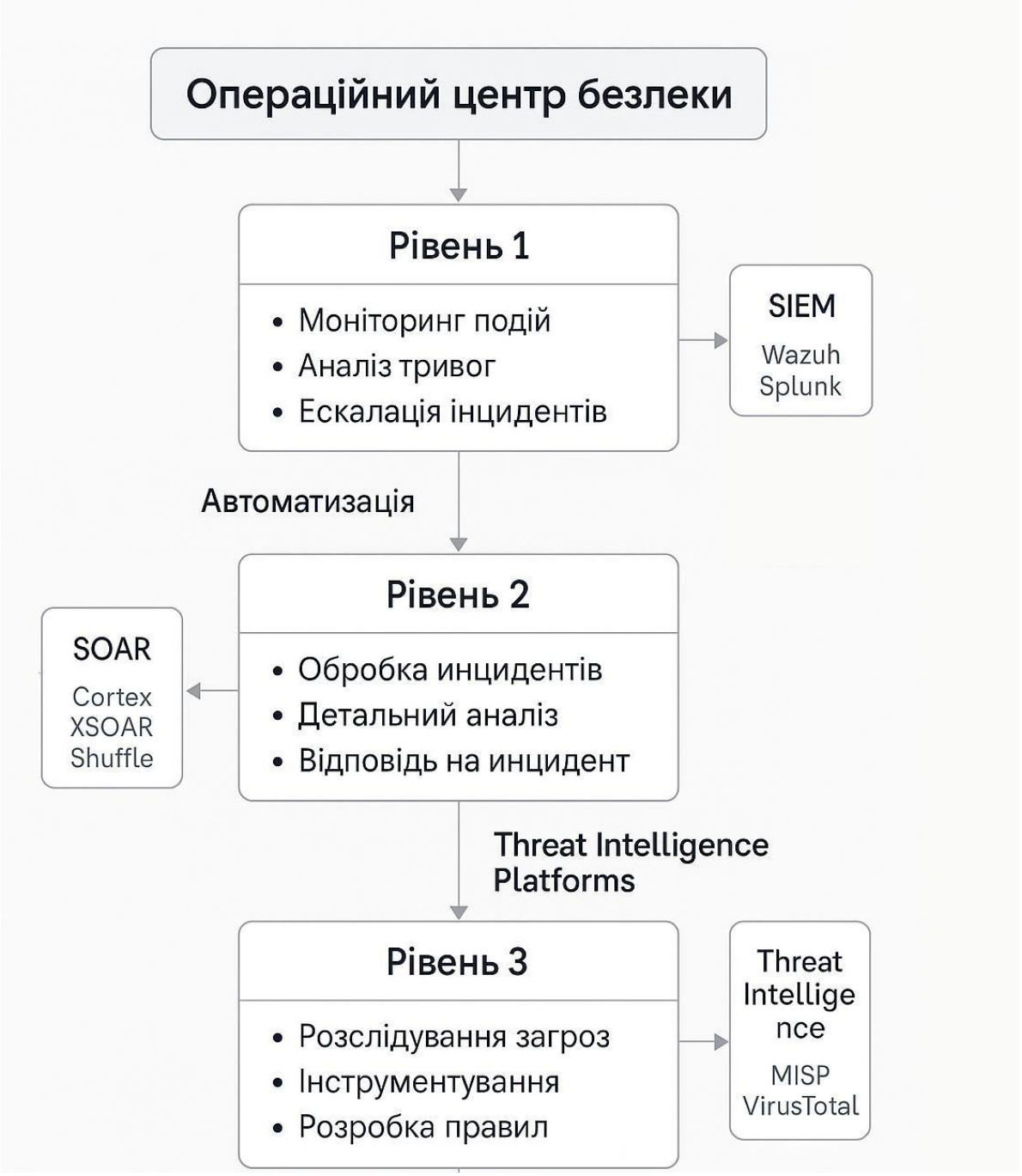
СХЕМА ВЗАЄМОДІЇ З ДЖЕРЕЛАМИ ДАНИХ



АЛГОРИТМ РОБОТИ ПРОГГРАМНОГО ЗАСТОСУНКУ



СТРУКТУРА ОПЕРАЦІЙНОГО ЦЕНТРУ БЕЗПЕКИ



ВИГЛЯД ЗГЕНЕРОВАНОГО ФАЙЛУ

звіт-summary.txt

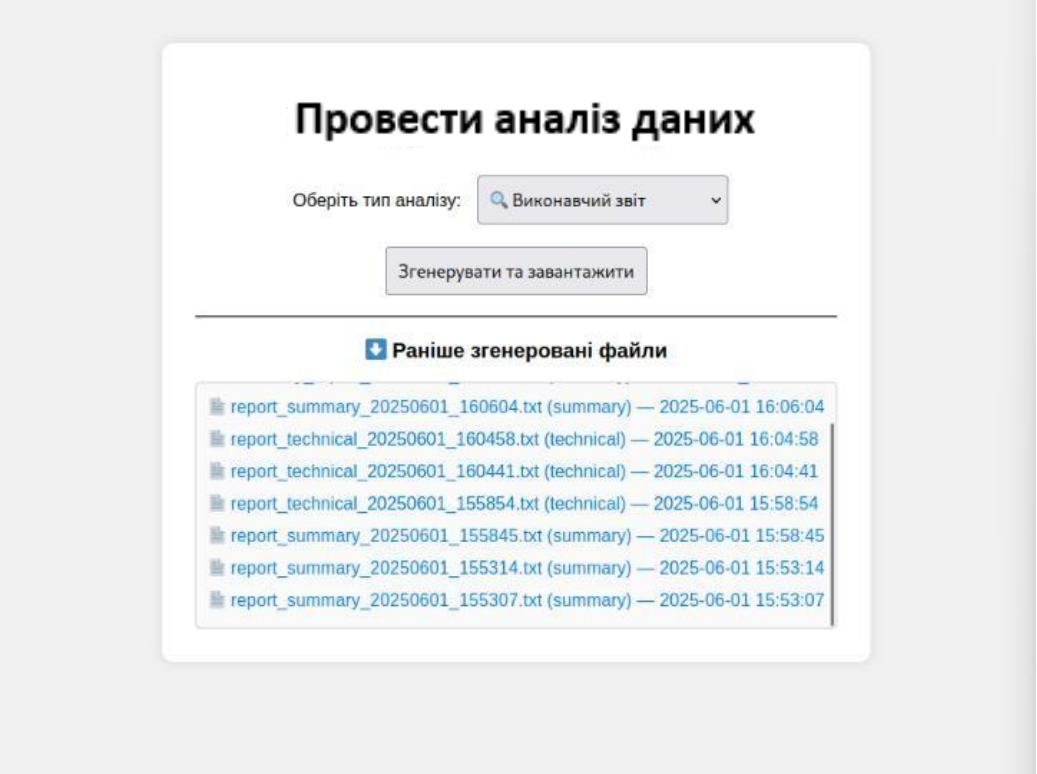
```
Було виявлено 366 інцидентів, серед яких 4 критичних. Один
хост має критичні інциденти.

**Топ 5 категорій інцидентів:**
1. PAM: Login session opened.
2. PAM: Login session closed.
3. Wazuh server started.
4. Wazuh agent started.
5. Wazuh agent disconnected.

**Топ 5 категорій критичних інцидентів:**
1. New user added to the system.
2. New group added to the system.
3. Successful sudo to ROOT executed.

**Топ 5 хостів по інцидентам:**
1. kali
2. wazuh.manager
```

ВИГЛЯД ВЕБ-ІНТЕРФЕЙСУ ПРОГРАМНОГО ЗАСОБУ



ФРАГМЕНТ ЗВІТУ ПРОГРАМНОГО ЗАСОБУ

звіт-summary.txt

```
Було виявлено 366 інцидентів, серед яких 4 критичних. Один
хост має критичні інциденти.

**Топ 5 категорій інцидентів:**
1. PAM: Login session opened.
2. PAM: Login session closed.
3. Wazuh server started.
4. Wazuh agent started.
5. Wazuh agent disconnected.

**Топ 5 категорій критичних інцидентів:**
1. New user added to the system.
2. New group added to the system.
3. Successful sudo to ROOT executed.

**Топ 5 хостів по інцидентам:**
1. kali
2. wazuh.manager
```