МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЧЕРКАСЬКИЙ ДЕРЖАВНИЙ ФАХОВИЙ БІЗНЕС-КОЛЕДЖ
КАФЕДРА КОМП'ЮТЕРНОЇ ІНЖЕНЕРІЇ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

на тему

МОДЕЛЮВАННЯ СЦЕНАРІЇВ ДЛЯ ПРИЙНЯТТЯ РІШЕНЬ У СИСТЕМАХ УПРАВЛІННЯ РИЗИКАМИ

Виконав: студент групи 1КІ-23
Спеціальності 123 «Комп'ютерна інженерія
Пилипенко М.С.
Керівник роботи
к.т.н., доцент Захарова М.В.
Кількість балів: _____
Оцінка: ECTS _____

Черкаси, 2025

ЧЕРКАСЬКИЙ ДЕРЖАВНИЙ БІЗНЕС-КОЛЕДЖ

Кафедра Комп'ютерної інженерії та інформаційних технологій

(повна назва випускової кафедри)

Спеціальність 123 "Комп'ютерна інженерія "

(шифр і назва спеціальності)

Освітня програма Комп'ютерна інженерія

(назва освітньої програми)

ЗАТВЕРДЖУЮ

Завідувач кафедри

комп'ютерної інженерії та інформаційних технологій

(назва кафедри)

_____ Хотунов В.І.

(підпис) (ПІБ)

«_____»_____2025 р.

З А В Д А Н Н Я

НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

Пилипенко Микола Сергійович

(прізвище, ім'я, по батькові студента)

1. Тема кваліфікаційної роботи Механізм підтримки прийняття рішень у кібербезпеці

Науковий керівник роботи Захарова Марія В'ячеславівна ,к.т.н., доцент

(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом закладу вищої освіти від "__" _____202_ року № __.

2. Строк подання студентом випускної роботи _____

3. Вихідні дані до випускної роботи Логи кіберзагроз із SIEM-систем, мережевий трафік, історичні дані про атаки, алгоритми машинного навчання, програмне забезпечення для моделювання та аналізу (OSSIM, Elastic Stack, Wireshark, Metasploit, Apache Kafka, Python).

4. Зміст випускної роботи (перелік питань, які потрібно розробити) У роботі будуть розглянуті методи управління ризиками та аналізу кіберзагроз, класифікація атак і сценарне моделювання загроз, застосування машинного навчання для прогнозування атак, інтеграція моделей аналізу в SIEM-системи, реалізація потокової обробки кіберінцидентів, а також тестування запропонованої системи та оцінка її ефективності.

5. Дата видачі завдання 15.09.2024р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Терміни виконання етапів	Примітка про виконання з підписами наукового керівника і студента
1	Вступ	13.10.2024	
2	Розділ 1. Підходи до управління ризиками в кібербезпеці	17.12.2024	
3	Розділ 2. Моделювання сценаріїв кібератак	07.03.2025	
4	Розділ 3. Розробка системи прийняття рішень на основі сценарного аналізу	09.04.2025	
5	Розділ 4. Тестування та оцінка ефективності розробленого підходу	07.05.2025	
6	Висновки	01.06.2025	
7	Оформлення випускної роботи (чистовий варіант)	03.06.2025	
8	Здача випускної роботи на кафедру для рецензування (за 14 днів до захисту)	05.06.2025	
9	Перевірка випускної роботи на наявність ознак плагіату (за 10 днів до захисту)	09.06.2025	
10	Подання випускної роботи на затвердження завідувачу кафедри (за 7 днів до захисту)	12.06.2025	

Студент _____ (підпис) Пилипенко М.С. (прізвище та ініціали)

Керівник роботи _____ (підпис) Захарова М.В. (прізвище та ініціали)

ПЕРЕЛІК ОСНОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ, СИМВОЛІВ І ОДИНИЦЬ, СПЕЦІАЛЬНИХ ТЕРМІНІВ, ЯКІ ВИКОРИСТОВУЮТЬСЯ

AI	(скор. від Artificial Intelligence) — штучний інтелект
API	(скор. від Application Programming Interface) — інтерфейс прикладного програмування
APT	(скор. від Advanced Persistent Threat) — високорозвинена постійна загроза
CPU	(скор. від Central Processing Unit) — центральний процесор
CSV	(скор. від Comma-Separated Values) — формат даних із розділенням комами
DDoS	(скор. від Distributed Denial of Service) — розподілена атака на відмову в обслуговуванні
DNS	(скор. від Domain Name System) — система доменних імен
DoS	(скор. від Denial of Service) — відмова в обслуговуванні
ELK	(скор. від Elastic Stack) — стек технологій Elasticsearch, Logstash, Kibana
ES	(скор. від Elasticsearch) — рушій пошуку та аналітики даних.
F1-score	— гармонічне середнє між точністю (precision) та повнотою (recall)
FP	(скор. від False Positive) — хибнопозитивне спрацьовування
FN	(скор. від False Negative) — хибнонегативне спрацьовування
IDS	(скор. від Intrusion Detection System) — система виявлення вторгнень
IoC	(скор. від Indicator of Compromise) — індикатор компрометації
IP	(скор. від Internet Protocol) — протокол інтернету
JSON	(скор. від JavaScript Object Notation) — формат обміну даними
Kafka	— система обробки потокових даних з відкритим кодом
ML	(скор. від Machine Learning) — машинне навчання

MITM	(скор. від Man-In-The-Middle) — атака типу "людина посередині"
NIDS	(скор. від Network Intrusion Detection System) — мережева система виявлення вторгнень
OS	(скор. від Operating System) — операційна система
Pandas	— бібліотека Python для аналізу та обробки табличних даних
Payload	— корисне навантаження, яке передається в рамках атаки
Precision	— точність класифікації
Python	— мова програмування, що використовується для аналітики, ML та автоматизації
QRadar	— комерційна SIEM-система від IBM
Recall	— повнота класифікації
SIEM	(скор. від Security Information and Event Management) — система управління подіями та інформацією безпеки
Scikit-learn	— бібліотека машинного навчання на Python
Snort	— система виявлення мережевих атак
SQL	(скор. від Structured Query Language) — мова запитів до баз даних
SSH	(скор. від Secure Shell) — протокол захищеного віддаленого доступу
SSL/TLS	— протоколи захищеного з’єднання
TP	(скор. від True Positive) — правильне позитивне спрацьовування
TN	(скор. від True Negative) — правильне негативне спрацьовування
TensorFlow	— бібліотека машинного навчання з відкритим кодом
VirtualBox	— програма для створення віртуальних машин
VM	(скор. від Virtual Machine) — віртуальна машина

ЗМІСТ

ВСТУП

Сучасне суспільство дедалі глибше інтегрується в цифровий простір. Інформаційні технології охопили майже всі сфери людської діяльності — від державного управління і медицини до освіти, фінансів та побуту. Цей процес супроводжується стрімким зростанням обсягів даних, які циркулюють у кіберпросторі, що, у свою чергу, створює нові виклики для забезпечення безпеки цієї інформації. З кожним роком зростає кількість кіберзагроз, які спрямовані як на окремих користувачів, так і на великі організації чи інфраструктурні об'єкти. Традиційні засоби захисту вже не завжди встигають за еволюцією загроз, тому виникає потреба в більш адаптивних та інтелектуальних методах забезпечення кібербезпеки. У цьому контексті особливої ваги набувають системи підтримки прийняття рішень, що дозволяють швидко оцінити ситуацію, виявити аномалії, запропонувати адекватні дії та мінімізувати ризики.

Актуальність. Розвиток цифрової економіки та глобалізація інформаційного обміну призвели до зростання важливості ефективного управління кібербезпекою. Виникає потреба в системах, які не тільки фіксують інциденти, але й допомагають приймати оптимальні рішення в умовах високої невизначеності та обмеженого часу. Механізми підтримки прийняття рішень дозволяють узагальнювати знання експертів, інтегрувати автоматизовані підходи та зменшити вплив людського фактору в ситуаціях ризику.

Мета і завдання роботи:

1. Дослідити зміст і значення механізмів підтримки прийняття рішень у сфері кібербезпеки.
2. Проаналізувати існуючі підходи до моделювання прийняття рішень у кіберзахисті.
3. Визначити критерії ефективності таких механізмів у контексті інформаційної безпеки.

4. Запропонувати модель або підхід до прийняття рішень, що може бути використаний для реагування на кіберінциденти.
5. Розглянути інструменти програмної реалізації рішень у сфері безпеки інформаційних систем.

Методи дослідження. Дослідження базується на загальнонаукових методах аналізу, синтезу, моделювання, а також застосуванні інструментів математичного апарату теорії прийняття рішень.

Об’єкт дослідження. Процес управління ризиками в інформаційній безпеці, включаючи виявлення, аналіз і реагування на кіберзагрози.

Предмет дослідження. Методи моделювання сценаріїв атак та механізми автоматизованого прийняття рішень у системах кібербезпеки на основі логів та потокових даних.

Практичне значення. Результати роботи можуть бути використані при проєктуванні та впровадженні адаптивних систем кіберзахисту, що здатні оперативно реагувати на загрози та зменшувати їх наслідки.

Постановка завдання. У роботі розглянуті методи управління ризиками та аналізу кіберзагроз, класифікація атак і сценарне моделювання загроз, застосування машинного навчання для прогнозування атак, інтеграція моделей аналізу в SIEM-системи, реалізація потокової обробки кіберінцидентів, а також тестування запропонованої системи та оцінка її ефективності.

Апробація результатів бакалаврської роботи. Основні результати та положення дослідження були представлені на студентських науково-практичних конференціях, а також під час проведення бесід з колегами.

Структура та обсяг роботи. Випускна кваліфікаційна робота складається зі вступу, чотирьох розділів, висновків, списку використаних джерел. Загальний обсяг роботи становить 75 сторінок основного тексту, 15 рисунків і 4 таблиці.

РОЗДІЛ 1 ПІДХОДИ ДО УПРАВЛІННЯ РИЗИКАМИ В КІБЕРБЕЗПЕЦІ

1.1 Теоретичні аспекти прийняття рішень у системах управління ризиками

1.1.1 Принципи та моделі оцінки ризиків у кібербезпеці

Оцінка ризиків є ключовим компонентом кібербезпеки, що дозволяє визначити ймовірність виникнення загроз і потенційні наслідки для інформаційних активів. Основна мета процесу оцінки ризиків полягає у зменшенні ймовірності реалізації кіберзагроз і мінімізації шкоди. Принципи оцінки ризиків передбачають системний підхід, регулярність, обґрунтованість, а також адаптивність до змін у зовнішньому та внутрішньому середовищі організації [1].

Серед основних моделей оцінки ризиків у кібербезпеці варто виокремити STRIDE — модель, що класифікує загрози за шістьма типами (спуфінг, модифікація, відмова, витік, відмова в обслуговуванні, привілеї), DREAD — яка оцінює ризик на основі п’яти критеріїв (шкода, відтворюваність, складність експлуатації, кількість користувачів, виявлення), а також FAIR — модель, яка кількісно аналізує ризики через фінансову перспективу, розраховуючи потенційні втрати [2].

Для реалізації оцінки ризиків на практиці організації часто застосовують стандарти ISO/IEC 27005, NIST SP 800-30, які формалізують такі етапи: ідентифікація активів, виявлення вразливостей, аналіз потенційних загроз, розрахунок ймовірностей та впливу, формування карти ризиків та пріоритетів. Поєднання кількісних і якісних підходів дозволяє отримати точнішу та релевантну інформацію для подальшого прийняття рішень у межах стратегії кіберзахисту [3].

1.1.2 Методи визначення критичних активів та рівнів загрози

Визначення критичних активів є одним із базових етапів побудови системи кібербезпеки. Критичні активи — це ті інформаційні, технологічні або бізнес-ресурси, порушення яких призводить до суттєвих наслідків для організації. Втрата таких активів може спричинити простої, фінансові втрати або репутаційні збитки [4].

Методи визначення критичних активів включають створення реєстру інформаційних ресурсів, оцінку їхньої значущості за параметрами конфіденційності, цілісності та доступності (CIA), побудову карти залежностей між елементами інфраструктури, а також класифікацію активів за критичністю відповідно до внутрішніх політик організації.

Рівень загрози визначається як функція ймовірності реалізації загрози та її впливу. Для цього застосовуються матриці ризику, сценарний аналіз інцидентів, статистичний аналіз логів і подій, а також виявлення індикаторів компрометації (IOC) у SIEM-системах. Автоматизовані засоби, такі як Elastic Stack, OSSIM або QRadar, дозволяють у реальному часі проводити подібну оцінку та оперативно реагувати на загрози.

1.1.3 Вплив динамічних загроз на процес оцінки ризиків

Сучасні кіберзагрози дедалі більше набувають динамічного характеру — змінюють вектори атак, адаптуються до контрзаходів, використовують новітні техніки, включаючи шифрування, соціальну інженерію, а також інструменти на основі штучного інтелекту. Такі загрози часто залишаються невидимими для традиційних систем виявлення, що значно ускладнює процес їхньої ідентифікації [5].

Щоб ефективно протистояти динамічним загрозам, процес оцінки ризиків має бути гнучким, безперервним і адаптивним. Доцільним є впровадження технологій поведінкового аналізу (UEBA), прогнозної аналітики, а також інтелектуальних механізмів виявлення нових шаблонів атак. Інформація, отримана з платформ обміну загрозами, таких як MISP чи

OTX, дозволяє оперативно оновлювати моделі ризику відповідно до актуальної картини загроз.

Таким чином, вплив динамічних загроз обумовлює необхідність модернізації класичних підходів до оцінки ризиків — від реактивних до проактивних та автоматизованих, здатних працювати в умовах постійної змінності середовища.

1.2 Класифікація атак і потенційних вразливостей

1.2.1 Види атак: мережеві, програмні та фізичні загрози

Класифікація кіберзагроз дозволяє краще розуміти логіку дій зловмисника, визначати можливі вектори атаки та оптимально розподіляти ресурси захисту. У загальному випадку атаки поділяються на три основні категорії: мережеві, програмні та фізичні.

Мережеві атаки відбуваються в межах інформаційних мереж і спрямовані на порушення їхньої доступності, конфіденційності або цілісності. До них належать атаки типу DoS/DDoS (див. рис 1.1), DNS spoofing, ARP poisoning, перехоплення сесій (session hijacking), атакуючі сканування портів, man-in-the-middle (MitM) (див. рис 1.2) тощо. Засоби протидії включають міжмережеві екрани, системи виявлення вторгнень (IDS), засоби шифрування трафіку, контроль доступу до мережевих сегментів [6].

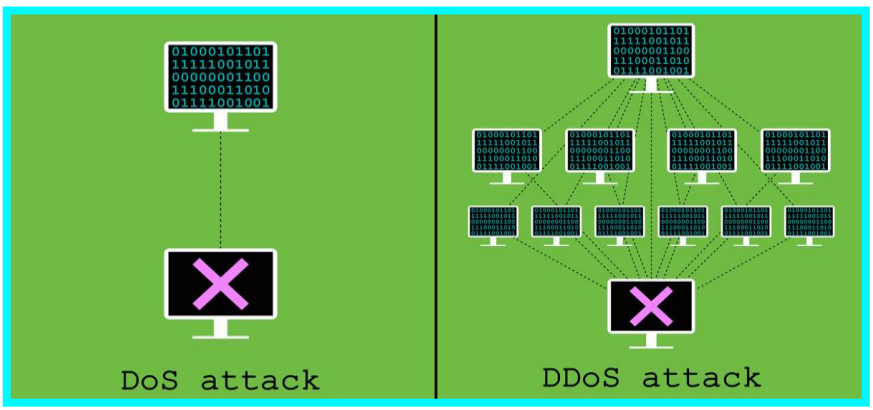

Рисунок 1.1 – Візуальний приклад роботи DoS/DDoS атак

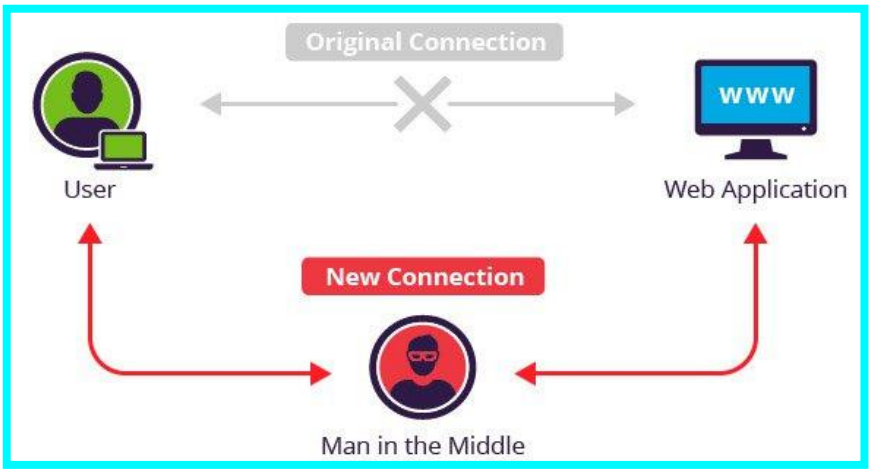

Рисунок 1.2 – Візуальна схема роботи MitM атак

Програмні атаки експлуатують вразливості у програмному забезпеченні або веб-застосунках. Найвідомішими є SQL-ін’єкції, XSS, буферні переповнення, віддалене виконання коду (RCE), а також атаки на API. Боротьба з такими атаками передбачає використання WAF, перевірку введених даних, дотримання принципів безпечної розробки (Secure Coding), оновлення ПЗ і проведення аудиту безпеки [7].

Фізичні загрози, хоча й менш поширені, є критичними: це крадіжки пристроїв, підключення шкідливих носіїв, встановлення кейлогерів, доступ до серверних кімнат. Захист здійснюється за допомогою відеоспостереження, контролю доступу, фізичного зонування, тривожних кнопок і політик обмеження фізичного доступу [8].

1.2.2 Основні типи вразливостей та методи їх експлуатації

Вразливості — це недоліки в системах, які зловмисники можуть використати для проведення атаки. Вони можуть виникати через помилки у програмному коді, неправильні конфігурації, людський фактор або застаріле обладнання. Вразливості класифікуються за джерелом (програмні, апаратні, конфігураційні), способом використання (локальні, віддалені), рівнем ризику (низький, середній, критичний).

Найбільш відомі типи вразливостей включають:

1. Небезпечні API (відсутність автентифікації або перевірки доступу).
2. Неправильне управління сесіями.
3. Небезпечне оброблення даних (наприклад, без перевірки введення).
4. Вразливості веб-застосунків, описані в OWASP Top 10.

Методи експлуатації варіюються залежно від типу вразливості: сканування портів, аналіз коду, ін’єкції, використання експлойтів. Зловмисники застосовують як готові фреймворки (наприклад, Metasploit), так і самостійно створені скрипти [9].

1.2.3 Соціальна інженерія та методи захисту від маніпуляцій

Соціальна інженерія базується на впливі на людину з метою отримання доступу до конфіденційної інформації. На відміну від технічних методів, соціальна інженерія використовує психологічні механізми довіри, страху, поспіху тощо.

До найбільш поширених методів належать:

1. Фішинг — надсилання повідомлень із підробленими посиланнями або вкладеннями.
2. Вішинг — телефонні дзвінки із соціальним тиском.
3. Смішинг — SMS-повідомлення із шкідливими посиланнями.
4. Передтекстинг — створення легенд для отримання інформації.

Захист полягає у підвищенні обізнаності співробітників, проведенні тренінгів, симуляціях фішингових атак, перевірках політик доступу. Також ефективним є впровадження двофакторної автентифікації, обмеження прав доступу, політик «мінімальних привілеїв» [10].

1.2.4 Використання штучного інтелекту для виявлення аномалій

Системи на основі штучного інтелекту (ШІ) дозволяють виявляти складні типи атак та аномалії, які важко зафіксувати традиційними методами. Такі системи аналізують поведінку користувачів, мережевий трафік, системні логи, а також контекст дій.

Найчастіше використовуються:

1. Алгоритми кластеризації (наприклад, K-means).
2. Методи навчання з підкріпленням.
3. Нейронні мережі для прогнозування поведінки.
4. Autoencoder-и для виявлення відхилень від норми.

У межах SIEM/UEBA-систем аналітика на базі ШІ дозволяє автоматизувати виявлення складних багатоступеневих атак (APT), блокувати підозрілу активність, формувати інтелектуальні правила реагування. Приклади платформ: Splunk, IBM QRadar, Sumo Logic (див. рис. 1.3) [11].

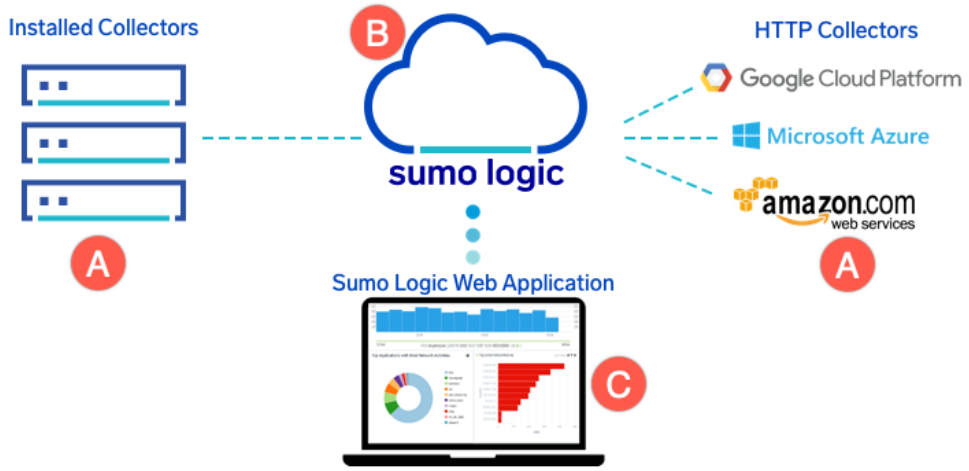

Рисунок 1.3 – Схема роботи платформ на прикладі Sumo Logic

Таким чином, поєднання знань про типи атак, вразливості, методи соціального впливу та інтелектуальні системи виявлення дозволяє формувати комплексну систему управління ризиками, здатну ефективно реагувати на сучасні кіберзагрози.

1.3 Використання OSSIM та Elastic Stack для моніторингу загроз

1.3.1 Архітектура та можливості OSSIM у кібербезпеці

Система OSSIM (Open Source Security Information Management) є одним із найвідоміших рішень з відкритим кодом у сфері інформаційної безпеки, що поєднує декілька інструментів для виявлення загроз, аналізу інцидентів та забезпечення централізованого моніторингу. Її головна особливість полягає у здатності інтегрувати в єдину систему такі компоненти, як Snort, OpenVAS, Suricata, OSSEC, Ntop та інші. Завдяки цьому користувач отримує універсальне середовище для збору, обробки та кореляції подій безпеки [12].

Архітектурно OSSIM базується на агентно-центричному підході, де на клієнтських вузлах встановлюються агенти (наприклад, OSSEC), які передають логи на центральний сервер. Сервер виконує кореляційний аналіз, зберігає дані та відображає їх у веб-інтерфейсі. Гнучкість OSSIM дозволяє налаштовувати власні правила кореляції, отримувати повідомлення про інциденти в режимі реального часу та формувати історичні звіти.

Важливо зазначити, що, попри безкоштовну ліцензію, OSSIM вимагає глибоких технічних знань для налаштування і підтримки. Здебільшого це пов’язано з великою кількістю інтегрованих компонентів, які мають власні конфігураційні параметри. Однак саме така гнучкість і функціональність робить OSSIM цінним інструментом для навчання та експериментальної побудови SIEM-рішень [13].

1.3.2 Аналіз логів і візуалізація загроз у Kibana

Elastic Stack, або так званий ELK Stack (Elasticsearch, Logstash, Kibana), є сучасним і масштабованим рішенням для централізованого збору, обробки, зберігання та візуалізації логів. У контексті кібербезпеки його застосування дозволяє не лише зберігати події з мережі та систем, але й ефективно аналізувати аномалії та створювати зручні візуальні інтерфейси для оцінки поточної ситуації.

У цій архітектурі Logstash виступає в ролі «розподільника» даних: він приймає інформацію з різних джерел (через Beats або інші вхідні модулі), обробляє її (нормалізує, фільтрує, збагачує) та передає в Elasticsearch. Останній забезпечує швидкий повнотекстовий пошук та аналітику. Kibana ж дає можливість створювати динамічні панелі з графіками, діаграмами та таблицями, які дозволяють швидко оцінити рівень загрози, частоту інцидентів або джерела атак (див. рис. 1.4).

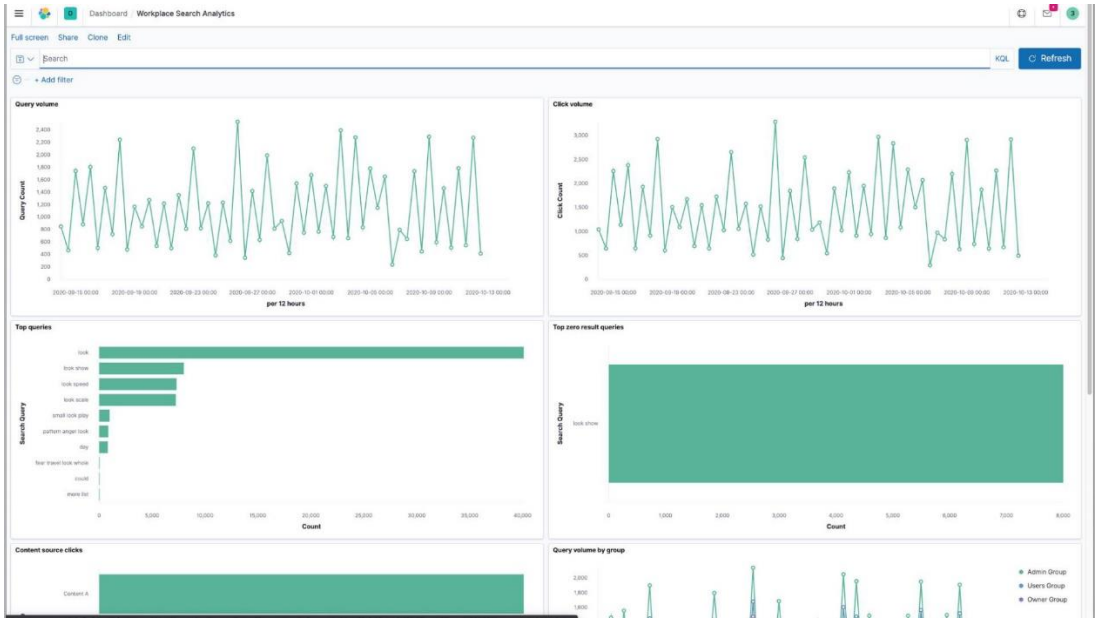

Рисунок 1.4 – Демонстрація вигляду робочого екрану Kibana

Перевагою Kibana є її інтуїтивно зрозумілий інтерфейс, що дозволяє навіть нетехнічним користувачам взаємодіяти з даними. Наприклад, можна в кілька кліків побудувати часову діаграму появи певної сигнатури або визначити IP-адреси, з яких найчастіше відбуваються підозрілі запити [14].

Інтеграція Kibana в систему безпеки значно покращує процес реагування на інциденти, оскільки надає візуальну картину подій, яка може бути зрозуміла навіть без глибокого аналізу логів вручну.

1.3.3 Порівняльний аналіз ефективності OSSIM та Elastic Stack

OSSIM і Elastic Stack мають різні архітектури та фокус, але обидва можуть бути ефективно використані для моніторингу кіберзагроз. OSSIM, як система класу SIEM, надає готову кореляційну логіку, яка здатна автоматично виявляти складні ланцюги атак. У той же час Elastic Stack пропонує більшу гнучкість у зберіганні та аналізі даних, а також має розширені можливості візуалізації. Для порівняння згідно критеріїв представлена відповідна таблиця 1.1.

Таблиця 1.1 - Порівняльний аналіз OSSIM та Elastic Stack

Критерій	OSSIM	Elastic Stack
Тип ліцензії	Відкрита	Відкрита + комерційні модулі
Можливості кореляції	Вбудовані	Потрібна інтеграція
Зручність	Високий поріг входу	Інтуїтивний інтерфейс Kibana
Масштабованість	Обмежена	Висока

У практичному застосуванні OSSIM зручний для невеликих організацій або лабораторій, які хочуть отримати базову функціональність без значних витрат. Elastic Stack же краще підходить для середніх і великих інфраструктур, де потрібна масштабованість, розподіленість і адаптація до нестандартних джерел логів.

Щодо продуктивності, OSSIM може поступатися за швидкістю обробки великих обсягів даних, особливо якщо система працює на базовому обладнанні. Elastic Stack, завдяки індексації в Elasticsearch, забезпечує високу продуктивність при пошуку навіть у великих наборах логів. Однак він не має

вбудованих механізмів кореляції подій, що потребує додаткових налаштувань або інтеграцій зі сторонніми системами.

У межах даної дипломної роботи OSSIM буде використовуватись для виявлення інцидентів, тоді як Elastic Stack — для візуального аналізу та побудови дашбордів. Це дозволить забезпечити комплексний підхід до моніторингу загроз та прийняття рішень [15].

1.4 Методи автоматичного виявлення загроз та прийняття рішень

1.4.1 Використання SIEM-систем для аналізу та реагування на інциденти

SIEM-системи (Security Information and Event Management) сьогодні є ключовим елементом у забезпеченні інформаційної безпеки підприємств, організацій і навіть державних установ. Основна цінність таких систем полягає в тому, що вони забезпечують централізований збір, обробку, збереження та аналіз подій з багатьох джерел інформації, таких як сервери, мережеве обладнання, програмні системи, бази даних та інші елементи інфраструктури. Завдяки функціям агрегації, нормалізації та кореляції логів SIEM дозволяє виявляти інциденти безпеки, які б інакше залишилися непоміченими.

Системи цього класу інтегруються з IDS/IPS-рішеннями, антивірусами, фаєрволами, проксі-серверами, контролерами домену тощо. Це дозволяє створити єдину картину поточного стану безпеки та вчасно виявляти аномалії. Наприклад, багаторазові невдалі спроби входу в систему, зміни прав доступу, підозрілі запити до бази даних — усе це може бути автоматично зафіксовано SIEM-системою, класифіковано як потенційна атака та передано для подальшого реагування. Застосування SIEM-систем дозволяє значно скоротити час між виявленням загрози та її нейтралізацією, що критично важливо в умовах сучасної швидкоплинної загрозової обстановки.

Крім виявлення, SIEM забезпечує можливості для реагування: надсилання повідомлень, блокування доступу, запуск скриптів. У поєднанні з системами автоматизації реагування (SOAR) SIEM перетворюється на активного учасника процесу захисту, а не лише спостерігача [16].

1.4.2 Роль машинного навчання в автоматичному виявленні загроз

Машинне навчання (ML) все більше інтегрується в інструменти кібербезпеки як технологія, що дозволяє вийти за межі традиційних підходів, заснованих на сигнатурах і жорстко заданих правилах. На відміну від класичних систем, які можуть виявити лише ті атаки, що вже були описані, моделі ML здатні навчатися на історичних даних, виявляти закономірності й відхилення, а також приймати рішення на основі нових, ще не ідентифікованих патернів.

Використання машинного навчання у виявленні загроз базується, зокрема, на методах класифікації та виявлення аномалій. Такі системи аналізують великі масиви логів і метаданих для того, щоб визначити, які події є типовими для конкретного середовища, а які — потенційно небезпечними. Наприклад, різке зростання запитів до певного API або поява підключень з нетипового географічного регіону можуть автоматично трактуватися як індикатор загрози.

Ключовою перевагою ML є його здатність адаптуватися до змін. Мережеве середовище, користувацька поведінка та самі методи атак постійно змінюються. Самонавчальні моделі дозволяють підтримувати ефективність захисту без постійного ручного оновлення правил. Більше того, сучасні SIEM-платформи дедалі частіше включають у себе модулі машинного навчання, які здатні підвищити точність спрацювань, зменшити кількість хибнопозитивних інцидентів та вивільнити ресурси фахівців із безпеки для складніших завдань [17].

1.4.3 Переваги та обмеження автоматизації процесу прийняття рішень

Автоматизація — один із найважливіших напрямів розвитку систем кіберзахисту. У великих компаніях обсяги подій, що генеруються щоденно, можуть досягати мільйонів записів. Аналіз такої кількості даних вручну є не лише неефективним, а й практично неможливим. Саме тому автоматизація процесів виявлення, класифікації та реагування на загрози є ключовим фактором ефективного захисту інформаційних систем.

Серед основних переваг автоматизації — швидкість прийняття рішень, можливість цілодобового моніторингу без участі людини, зниження ризику помилок, що виникають через людський фактор, а також скорочення витрат на обслуговування великої кількості інцидентів. Наприклад, при надходженні підозрілої події система може автоматично перевірити її за базою відомих атак, заблокувати доступ, повідомити відповідального спеціаліста або навіть ініціювати процедуру сканування інфраструктури на наявність інших загроз.

Втім, слід визнати, що автоматизація має і свої обмеження. Найбільш поширеним є ризик хибнопозитивних спрацювань, коли система помилково ідентифікує легітимну активність як загрозу. У результаті можуть бути заблоковані звичайні користувачі або зупинені важливі бізнес-процеси. Крім того, автоматизовані рішення потребують ретельного налаштування, тестування і регулярного оновлення. Без цього вони можуть стати неефективними або навіть шкідливими [18].

Найбільш ефективною є комбінована модель, в якій автоматизація використовується для виконання рутинних операцій, а кінцеве рішення щодо критичних інцидентів приймає фахівець з інформаційної безпеки.

1.4.4 Використання Apache Kafka для потокового аналізу загроз

Apache Kafka є одним з найпотужніших інструментів потокової обробки даних, який набув широкого застосування в сфері кібербезпеки. Завдяки своїй архітектурі Kafka дозволяє у реальному часі отримувати, зберігати і передавати великі обсяги логів та подій із різних джерел — від серверів і

мережевого обладнання до кінцевих пристроїв і додатків. У середовищі, де час виявлення загроз має критичне значення, можливість обробляти події одразу після їх надходження дає відчутну перевагу.

Kafka функціонує як посередник між джерелами даних і системами обробки. Наприклад, агенти Filebeat можуть надсилати логи до Kafka, а далі події спрямовуються в Logstash, де здійснюється їх фільтрація, нормалізація та підготовка до аналізу. Це дає змогу значно зменшити навантаження на кінцеві системи зберігання та аналізу, а також забезпечити масштабованість [19].

Завдяки своїй модульності та гнучкості Kafka легко інтегрується з різноманітними системами: Elastic Stack, Apache Flink, Spark, Hadoop тощо. Це дозволяє будувати потужні архітектури для виявлення загроз, поведінкової аналітики, прогнозування ризиків і автоматизованого реагування. Особливо актуальною є можливість розділення тем (topics), що дозволяє організовувати потоки даних за джерелами або типами інцидентів, оптимізуючи обробку та аналіз.

В контексті роботи Apache Kafka може бути використана як ключова шина обміну повідомленнями між компонентами системи, що дасть змогу забезпечити швидку реакцію на загрози та гнучкість розгортання.

РОЗДІЛ 2 МОДЕЛЮВАННЯ СЦЕНАРІЇВ КІБЕРАТАК

2.1. Методи аналізу кіберзагроз за допомогою Python

2.1.1 Обробка великих обсягів логів за допомогою Pandas

У сучасних умовах забезпечення кібербезпеки одним із найважливіших джерел інформації про інциденти є журнали подій, або логи. Вони містять величезну кількість даних, які можуть використовуватись для виявлення підозрілої активності, відстеження поведінки користувачів і аналізу атак. Проте через свій обсяг та неструктурованість ці дані потребують ефективного підходу до обробки. У цьому контексті бібліотека Pandas у Python виступає потужним інструментом для зчитування, попередньої обробки та агрегації логів.

Pandas дозволяє завантажувати лог-файли з різних форматів (CSV, JSON, TSV тощо), структурувати їх у вигляді таблиць (DataFrame) та виконувати фільтрацію, сортування, групування, обчислення статистичних показників. Особливо корисними функціями є «.groupby()», «.pivot_table()», «.resample()», які дозволяють швидко побачити динаміку подій у часі, визначити частоту виникнення певних типів подій, джерела трафіку чи IP-адреси, з яких здійснюється активність.

У рамках обробки логів Pandas також допомагає очистити дані, наприклад, видалити дублі, пропущені значення або невірні формати. Крім того, за допомогою бібліотеки datetime логічно поєднується часовий контекст, що є критично важливим при аналізі інцидентів, які відбуваються у вигляді послідовностей подій.

Використання Pandas дозволяє значно скоротити час попередньої обробки логів, що робить його основою для подальшої аналітики, зокрема для застосування статистичних методів чи машинного навчання [20].

2.1.2 Методи статистичного аналізу для виявлення аномалій

Після попередньої обробки логів виникає необхідність аналізу поведінки системи зRan метою виявлення аномальних подій, які можуть свідчити про зловмисну активність. Статистичний аналіз дає змогу виявити ті випадки, які відхиляються від «нормальної» поведінки та потребують додаткової уваги.

Одним із базових підходів є аналіз розподілу значень ключових полів логів: кількість запитів з одного IP-адресу, частота звернень до певних портів, час відповіді серверу тощо. За допомогою середнього значення (mean), медіани (median), стандартного відхилення (std), коефіцієнта асиметрії та інших статистичних показників можна визначити типові параметри для нормальної поведінки системи.

Коли певні значення виходять за межі визначених порогів (наприклад, ±3σ від середнього), це може розглядатися як потенційна аномалія. Для визначення наскільки значення відхиляється від середнього, використовується формула Z-оцінки (2.1):

$$Z = \frac{X - \mu}{\sigma}$$

(2.1),

де X — значення ознаки;

μ — середнє значення;

σ — стандартне відхилення.

Такі події можуть включати: велику кількість логінів за короткий проміжок часу, спроби доступу до адміністративних ресурсів з незвичних адрес, зміни конфігурацій без підтвердження [21].

Також важливо аналізувати частотні закономірності — як часто і з яких джерел надходять події, чи є різкі піки в активності тощо. У Python це можна реалізувати за допомогою Pandas, NumPy або бібліотеки SciPy, що дозволяє

застосовувати статистичні тести: наприклад, критерій Шапіро-Уїлка для перевірки нормальності, або тест Зіглера–Нікольса для зміни середнього рівня.

У поєднанні з візуалізацією даних (через Matplotlib або Seaborn) статистичний аналіз створює потужну основу для формування перших рівнів сценаріїв автоматичного виявлення аномалій, ще до застосування машинного навчання.

2.1.3 Алгоритми класифікації загроз у Scikit-learn

Після виявлення аномалій за допомогою статистики виникає потреба класифікувати події — відрізнити нормальну поведінку від загрозливої та визначити тип інциденту. У цьому контексті доцільно застосовувати алгоритми машинного навчання, зокрема ті, що реалізовані у бібліотеці Scikit-learn — одному з найпоширеніших інструментів у сфері прикладного аналізу даних.

Scikit-learn пропонує набір моделей для класифікації: логістична регресія, дерево рішень, метод опорних векторів (SVM), наївний баєсівський класифікатор, випадкові ліси (Random Forest) та інші. Формула Gini Index застосовується як критерій розгалуження у випадку дерева рішень(2.2):

$$G = 1 - \sum_{i=1}^{n} p_i^2$$

(2.2),

де p_i — частка елементів класу i в наборі даних;

$\sum$ — знак суми за всіма класами.

Вибір алгоритму залежить від типу даних та задачі. У випадку аналізу логів часто використовують багатокласову або бінарну класифікацію: наприклад, подія може бути класифікована як нормальна, підозріла або шкідлива [22].

Моделі машинного навчання будуються на підставі тренувального набору даних, що містить приклади відомих подій із позначеними класами. Для цього попередньо проводиться фічеризація — перетворення логів на вектори ознак: кількість спроб входу, час між запитами, розмір трафіку тощо. Далі виконується навчання моделі та оцінка її точності за метриками: точність (accuracy), повнота (recall), точність класифікації (precision), F1-міра.

Метрики класифікації наведено в таблиці 2.1. Найкращі результати показала модель Random Forest за всіма основними показниками.

Таблиця 2.1 - Метрики класифікації моделей

Модель	Precision	Recall	F1-score	Accuracy
Decision Tree	0.89	0.83	0.86	0.87
Random Forest	0.93	0.88	0.90	0.91
SVM	0.88	0.79	0.83	0.85

Після навчання модель можна застосовувати до потоку нових подій — система автоматично класифікує їх і вказує рівень загрози. Це дозволяє реалізувати інтелектуальний рівень у системі прийняття рішень: наприклад, події, класифіковані як критичні, одразу направляються на обробку спеціалістом або блокуються автоматично.

Таким чином, Scikit-learn у поєднанні з Pandas і попереднім статистичним аналізом дозволяє реалізувати повноцінний цикл обробки логів — від збору до інтелектуальної класифікації, що є основою для автоматизованого управління кіберризиками [23].

2.2 Використання TensorFlow для прогнозування атак

2.2.1 Архітектура глибоких нейронних мереж для кібербезпеки

Глибокі нейронні мережі (Deep Neural Networks, DNN) демонструють високий потенціал у виявленні складних шаблонів у великих масивах даних, зокрема в сфері кібербезпеки. Їх використання дозволяє моделювати поведінкові аномалії, передбачати розвиток атак і виявляти загрози, які неможливо виявити звичайними сигнатурними або статистичними методами. Фреймворк TensorFlow, розроблений Google, надає широкі можливості для побудови, навчання та впровадження глибоких моделей у реальних інформаційних системах.

Архітектура таких мереж може включати багатошарові перцептрони, згорткові нейронні мережі (CNN), рекурентні мережі (RNN) та їхні модифікації — зокрема LSTM (Long Short-Term Memory), які добре підходять для аналізу послідовностей подій, характерних для логів безпеки. Рекурентні структури особливо корисні при роботі з часовими рядами — наприклад, у виявленні послідовностей дій, що передують атаці типу "ланцюжок дій" [24].

Типова глибока модель для виявлення загроз включає вхідний шар, що приймає багатовимірні ознаки (кількість запитів, тривалість сесій, IP-геолокація тощо), кілька прихованих шарів із функціями активації (наприклад, ReLU), механізми нормалізації та регуляризації, і вихідний шар, який формує прогноз або ймовірність інциденту. Важливою особливістю TensorFlow є можливість оптимізувати структуру моделі відповідно до складності задачі, використовувати GPU-прискорення та легко експортувати моделі для подальшого застосування у продуктивному середовищі.

2.2.2 Навчання моделей на основі історичних загроз

Ефективність прогнозної моделі в кібербезпеці напряму залежить від якості навчальних даних. У випадку з TensorFlow, процес навчання передбачає формування вибірки подій, які вже були класифіковані у минулому як загрози

або безпечна активність. Така історична інформація зазвичай формується на основі логів систем безпеки, архівів SIEM-систем, даних IDS/IPS або джерел із загальнодоступних кіберінцидентів [25].

Перед початком навчання дані проходять етап попередньої обробки, який включає очищення від шуму, нормалізацію числових значень, перетворення категоріальних змінних у числові представлення (one-hot encoding, label encoding), а також балансування вибірки, що є особливо важливим, оскільки загрозливі події часто складають меншість. Недобалансованість класів може призвести до зміщення моделі в бік безпечної поведінки й знизити її здатність виявляти реальні атаки.

Після підготовки даних виконується розбиття на тренувальний, валідаційний та тестовий набори. У процесі навчання застосовується метод зворотного поширення помилки (backpropagation) з оптимізацією вагових коефіцієнтів за допомогою градієнтного спуску або його модифікацій (наприклад, Adam, RMSprop). TensorFlow надає інструменти для відстеження метрик точності, втрат, а також графічного відображення динаміки навчання.

Історичні дані також можуть використовуватись для побудови віконних моделей, у яких на вхід подається певна кількість попередніх подій. Такий підхід дає змогу моделі не лише оцінювати поточний стан системи, а й враховувати її попередню поведінку, що значно підвищує точність прогнозів [26].

2.2.3 Аналіз ефективності прогнозування та мінімізація помилкових спрацьовувань

Якість моделі прогнозування атак не визначається лише її здатністю правильно класифікувати загрози. У практичному застосуванні не менш важливим показником є рівень помилкових спрацьовувань — тобто випадків, коли нормальна активність визначається як підозріла (false positive) або загроза — як нешкідлива (false negative). Високий рівень хибнопозитивних

спрацювань створює додаткове навантаження на операторів безпеки, тоді як хибнонегативні можуть залишити атаку непоміченою.

Оцінка ефективності виконується за допомогою метрик: точність (accuracy), повнота (recall), точність класифікації (precision), F1-міра, AUC-ROC крива. При цьому залежно від задачі може бути важливішим мінімізувати false negatives (щоб не проґавити реальні атаки), навіть якщо при цьому буде більше false positives. Підбір оптимального балансу — це завжди компроміс між навантаженням на систему та безпекою [27].

TensorFlow дозволяє контролювати значення функцій втрат, ваги класів, техніки регуляризації (L2, Dropout), що дає змогу впливати на поведінку моделі. Також застосовуються методи тонкої настройки (fine-tuning), ансамблі моделей, механізми attention та перевірка на наборах даних, які не брали участь у навчанні. Також використовується функція Loss function, яка показує, як оптимізується модель при тренуванні(2.3):

$$Loss = -\sum_{i-1}^{n} y_i \times \log(\hat{y_i})$$

(2.3),

де y_i — істинне значення (мітка класу);

$\hat{y}_i$ — передбачене значення моделі;

$\sum$ — сума по всіх об’єктах у вибірці.

Для зменшення помилкових спрацьовувань важливо також проводити інтерпретацію рішень моделі. З цією метою можуть використовуватись інструменти пояснення моделей, як-от SHAP або LIME, які дозволяють зрозуміти, які саме ознаки вплинули на прогноз. Це підвищує довіру до системи, а також допомагає в оптимізації сценаріїв реагування.

2.2.4 Інтеграція прогнозної аналітики в SIEM-системи

Прогнозна аналітика, заснована на глибокому навчанні, має повноцінну практичну цінність лише за умови її інтеграції в існуючі системи моніторингу й управління безпекою. У цьому контексті важливим є не лише створення моделі, але й організація її зв’язку з джерелами даних, інтерфейсами управління інцидентами, а також забезпечення її роботи в реальному часі [28].

Інтеграція моделей TensorFlow в SIEM-системи може здійснюватися кількома шляхами. Один із них — це використання REST API, через який модель доступна як окрема служба: вона приймає на вхід події у форматі JSON та повертає прогноз. Такий варіант дозволяє легко масштабувати рішення та розгортати кілька моделей одночасно. Інший варіант — використання сервісів потокової обробки, таких як Apache Kafka або Flink, які передають дані безпосередньо з агентів (наприклад, Filebeat) до моделі, минаючи проміжне збереження.

У SIEM-системі аналітика може використовуватись на різних рівнях: як модуль попередньої фільтрації, що допомагає зменшити кількість подій, які розглядаються аналітиками; як компонент сценарного реагування, що активується при виявленні певної поведінки; або як частина кореляційного движка, який оцінює ризики на основі множини сигналів.

Одним із викликів інтеграції є забезпечення продуктивності й стабільності. Моделі машинного навчання потребують ресурсів, тому важливо оптимізувати їх, використовувати стислі формати представлення, зменшувати складність моделей без втрати точності. У цьому можуть допомогти сервіси TensorFlow Lite або TensorFlow Serving [29].

Загалом, інтеграція прогнозної аналітики в SIEM відкриває новий рівень адаптивного захисту, де система не лише реагує на інциденти, а й передбачає їх, даючи змогу вчасно вжити заходів і мінімізувати ризики.

2.3 Побудова сценаріїв атак на основі історичних даних

2.3.1 Аналіз зібраних даних та визначення основних шаблонів атак

Історичні дані є цінним джерелом для розуміння того, як саме відбуваються кіберінциденти. Логи, записи з SIEM-систем, систем виявлення вторгнень (IDS), результати аудиту — усе це створює інформаційну основу для виявлення поведінкових шаблонів, що властиві певним типам атак. Аналіз таких даних дозволяє відновити послідовність дій зловмисника, визначити типові точки входу, методи ескалації привілеїв, способи пересування мережею (lateral movement) та техніки уникнення виявлення.

Основна мета аналізу полягає в тому, щоб виокремити ключові характеристики подій, які найчастіше передують або супроводжують атаки. До таких характеристик належать: повторюваність дій у певні часові періоди, фіксація однакових IP-адрес у різних інцидентах, активність у нетиповий час, використання незвичних портів чи протоколів, послідовне змінення атрибутів доступу тощо. За допомогою інструментів Pandas, NumPy, SQL-запитів або спеціалізованих аналітичних платформ проводиться попередня агрегація, нормалізація та сегментація даних для побудови повної картини поведінки загрози [30].

На підставі зібраних даних формуються патерни атак — повторювані послідовності дій, які можна потім використовувати для ідентифікації схожих ситуацій у майбутньому. Ці шаблони можуть стати основою для сценарного моделювання атак, яке застосовується в аналітичних, дослідницьких та практичних цілях — зокрема, для навчання аналітиків, розробки автоматизованих механізмів реагування або перевірки стійкості систем до нових типів загроз.

2.3.2 Використання ланцюгів Маркова для прогнозування розвитку інцидентів

Ланцюги Маркова є потужним математичним інструментом для моделювання стоячих імовірнісних процесів, де поточний стан системи залежить лише від попереднього. У сфері кібербезпеки це означає, що кожна подія — така як успішний вхід у систему, зміна привілеїв, спроба виконання коду — розглядається як окремий стан, а перехід між ними має певну ймовірність. Таким чином, можна побудувати модель, яка відображає логіку розвитку атак, засновану на реально зафіксованих послідовностях подій [31].

Побудова моделі на основі ланцюгів Маркова передбачає виділення унікальних станів на підставі логів або сценаріїв, які були зібрані та структуровані на попередньому етапі. Кожен перехід між станами аналізується з точки зору частоти його виникнення. В результаті формується матриця переходів, у якій зазначено, з якою ймовірністю система переходить від одного стану до іншого. Цей підхід дозволяє імітувати поведінку атакувальника, прогнозувати можливі варіанти розвитку подій і виявляти критичні точки, де захист повинен бути особливо пильним.

Матриця ймовірностей ланцюгів Маркова допомогає візуалізувати ймовірності переходів між станами в сценаріях атак (2.4):

$$P = \begin{bmatrix} 0.6 & 0.4 \\ 0.3 & 0.7 \end{bmatrix} \quad (2.4),$$

де P — матриця переходів у ланцюзі Маркова;
значення вказують ймовірності переходу між станами.

Застосування ланцюгів Маркова корисне не лише для аналізу вже відомих атак, а й для прогнозування майбутніх. Наприклад, якщо виявлено, що після певного типу запитів часто слідує спроба зміни конфігурації системи або доступу до бази даних, то система може заздалегідь сформувати відповідний

сценарій реагування. Модель також дозволяє визначити вірогідність завершення атаки успішним проникненням або її перериванням, що допомагає розставити пріоритети у моніторингу та розподілі ресурсів [32].

2.3.3 Генерація сценаріїв атак на основі отриманих патернів

Після виявлення повторюваних шаблонів поведінки та побудови моделі ймовірнісних переходів постає завдання створити сценарії атак, які можуть бути використані для тестування системи, симуляції інцидентів або автоматизованого реагування. Сценарій у цьому контексті — це логічна послідовність дій, що відтворює характерну атаку, з урахуванням її ймовірного розвитку та результату.

Генерація таких сценаріїв передбачає комбінацію історичних даних, поведінкових моделей та елементів прогностичної аналітики. Кожен сценарій включає початкові умови (наприклад, входження у систему під обліковим записом адміністратора), серію кроків, що включають взаємодію з мережею, використання вразливостей, виконання команд, та кінцеву ціль — наприклад, ексфільтрацію даних або захоплення контролю над вузлом. На основі ймовірностей, визначених через ланцюги Маркова, сценарій може мати декілька варіантів розвитку — від найтиповішого до менш вірогідного, але потенційно небезпечного [33].

Такі сценарії мають важливе значення в практиці тестування безпеки — наприклад, у рамках пентестів, red team/blue team симуляцій або моделювання інцидентів у режимі навчання. Вони дозволяють перевірити стійкість систем до реалістичних атак, оцінити ефективність існуючих заходів реагування, а також виявити слабкі місця в архітектурі безпеки.

Більш того, згенеровані сценарії можуть бути безпосередньо інтегровані в SIEM або SOAR-системи як модулі автоматизованого аналізу. Це дозволяє перевести сценарії з категорії навчального інструменту до реального засобу захисту, що працює в середовищі, близькому до бойового.

2.4 Аналіз мережевого трафіку у Wireshark для виявлення загроз

2.4.1 Використання фільтрів для виявлення підозрілих пакетів

Wireshark є одним із найпотужніших і найпоширеніших інструментів для аналізу мережевого трафіку. Його головна перевага полягає у здатності здійснювати глибоку інспекцію кожного пакета, що проходить через мережу. Для виявлення потенційно шкідливої активності важливим компонентом роботи з Wireshark є застосування фільтрів, які дозволяють зосередитись лише на релевантних для безпеки даних [34].

Фільтри у Wireshark працюють на рівні захоплення (capture filters) та на рівні відображення (display filters). Захоплення дозволяє обмежити обсяг даних, які записуються у файл, що особливо актуально для високонавантажених мереж. Натомість фільтри відображення використовуються для глибшого аналізу вже зібраного трафіку.

Зокрема, аналітик може налаштувати фільтри для виявлення специфічних типів трафіку: наприклад, підозрілих HTTP-запитів, пакетів ICMP з аномальною частотою, трафіку на незвичні порти або пакети з ознаками сканування. Крім того, Wireshark дозволяє комбінувати фільтри за допомогою логічних операторів, що відкриває широкі можливості для побудови складних умов виявлення загроз.

Застосування таких фільтрів суттєво підвищує ефективність аналітики, дозволяє швидше виявляти потенційні атаки та мінімізувати кількість хибних спрацьовувань [35].

2.4.2 Дослідження DDoS-атак та несанкціонованих підключень

Одним із поширених сценаріїв застосування Wireshark у практиці інформаційної безпеки є аналіз розподілених атак типу DDoS, а також виявлення несанкціонованих спроб доступу до системи. У таких випадках інструмент дозволяє не лише фіксувати факт аномальної активності, але й деталізовано дослідити її природу.

DDoS-атаки зазвичай супроводжуються великою кількістю однотипних запитів до певного сервісу або порту. У Wireshark це проявляється у вигляді повторюваних пакетів з однаковою структурою, що надходять із великої кількості IP-адрес або навпаки — з одного джерела, але в надмірній кількості. Аналіз часу між пакетами, TTL, джерел трафіку та протоколів дозволяє ідентифікувати природу навантаження: чи йдеться про SYN-флуд, UDP-флуд чи HTTP GET-спам.

Несанкціоновані підключення, зі свого боку, можуть бути виявлені через спроби встановлення сесій поза межами робочих годин, підключення до служб, які не використовуються, або передачу незашифрованих облікових даних. За допомогою Wireshark можна прослідкувати спроби аутентифікації, відстежити використовувані протоколи та виявити незвичні з'єднання з іноземних геолокацій.

Ці типи атак залишають виразні цифрові сліди, які при детальному аналізі дають змогу розпізнати їх ще до того, як буде завдано шкоди [36].

2.4.3 Аналіз протоколів та їх роль у кіберзагрозах

Значну частину аналітики у Wireshark становить вивчення специфіки протоколів, які використовуються у мережевій взаємодії. Багато сучасних атак реалізуються через легітимні канали зв’язку — HTTP, DNS, SMB, FTP, що ускладнює їх виявлення. Розуміння особливостей роботи кожного з цих протоколів дозволяє краще виявляти аномалії у їх використанні.

Наприклад, DNS-запити зазвичай короткі й передбачувані. Якщо фіксується надмірна кількість нестандартних запитів або довгі піддоменні імена — це може вказувати на спробу ексфільтрації даних через DNS-тунелювання. У протоколі HTTP аналіз структури заголовків, User-Agent, методів запитів (GET, POST, PUT) дозволяє виявити ознаки сканування або експлуатації вразливостей.

Особливу увагу слід звертати на використання незашифрованих протоколів, таких як Telnet або FTP, які зловмисники можуть використовувати

для підключення до систем з мінімальним рівнем захисту. Також аномальною вважається зміна характеру трафіку — наприклад, передача бінарних даних через канали, які зазвичай використовуються для текстових запитів [37].

Wireshark дозволяє досліджувати кожен протокол на глибокому рівні, включаючи вміст полів заголовків, флагів управління, розміри пакетів, а також порядок і напрямок їх передачі.

2.4.4 Інтеграція результатів аналізу у систему моніторингу

Для того, щоб аналіз у Wireshark мав не лише діагностичну, але й практичну цінність, його результати повинні інтегруватися в більш масштабні системи моніторингу й реагування. Це може бути як частина ручного аналізу аналітиком SOC, так і автоматизоване передавання інформації до SIEM або інструментів оркестрації (SOAR).

Wireshark дозволяє експортувати дані у різних форматах — CSV, JSON, PCAP, XML, які можуть бути оброблені скриптами Python або імпортовані у системи на базі Elasticsearch, Kibana, Splunk тощо. Зокрема, зібрані та відфільтровані пакети можуть бути перетворені у події, на які система реагує згідно з визначеними сценаріями. Це дозволяє розширити класичну функціональність SIEM-систем за рахунок глибшого рівня аналізу, що виконується окремими фреймами трафіку [38].

Крім того, в середовищах із підвищеним ризиком Wireshark може бути використаний як частина автоматизованої системи — наприклад, у комбінації з IDS типу Suricata або Zeek, де результати аналізу трафіку можуть бути збагачені (enriched) та передані до аналітичного ядра. Таким чином досягається повноцінний цикл: виявлення → аналіз → реагування, із використанням як сигнатурного, так і поведінкового підходів.

РОЗДІЛ 3 РОЗРОБКА СИСТЕМИ ПРИЙНЯТТЯ РІШЕНЬ НА ОСНОВІ СЦЕНАРНОГО АНАЛІЗУ

3.1 Архітектура системи прийняття рішень у контексті SIEM

3.1.1 Взаємодія модулів аналізу загроз із SIEM-системами

Сучасні SIEM-системи відіграють ключову роль у процесі моніторингу, виявлення та реагування на інциденти інформаційної безпеки. Проте ефективність їх роботи значно підвищується при інтеграції з додатковими модулями, що виконують аналітичні, прогностичні чи поведінкові функції. Така взаємодія дозволяє перейти від базового збору й кореляції подій до складної архітектури прийняття рішень, заснованої на даних, аналізі закономірностей і навчанні моделей.

Основна мета взаємодії полягає в тому, щоб забезпечити двосторонній обмін даними між SIEM та зовнішніми аналітичними модулями. З одного боку, SIEM надає структуровані лог-файли, події з агентів, IDS/IPS, фаєрволів, AV-рішень тощо. З іншого — зовнішні модулі (наприклад, моделі на базі TensorFlow або Scikit-learn) аналізують ці дані та повертають класифіковані події, ймовірності ризику чи конкретні рішення щодо реагування [39].

Такі модулі можуть бути реалізовані як окремі мікросервіси, які взаємодіють через REST API, або як вбудовані скрипти в середовищі самого SIEM (наприклад, через Logstash pipelines або custom enrichment плагіни). Ключовим елементом архітектури є механізм передачі результатів назад у SIEM — наприклад, через Kafka topic, Elastic index або syslog, після чого SIEM відображає аналітичні висновки в дашбордах чи генерує інциденти.

Завдяки такій взаємодії SIEM-система набуває здатності працювати не лише реактивно, а й проактивно, використовуючи динамічні правила та оновлювані моделі поведінки для оцінки загроз у режимі реального часу.

3.1.2 Інтеграція системи прогнозування у реальні інфраструктури

Інтеграція прогнозних систем у реальні IT-інфраструктури вимагає комплексного підходу, який враховує як технічні, так і організаційні аспекти. Система прогнозування має відповідати вимогам до продуктивності, безпеки, сумісності з існуючими компонентами, а також забезпечувати масштабованість і стійкість до відмов [40].

Процес інтеграції зазвичай починається з аналізу існуючої архітектури підприємства: визначення точок збору даних (агенти, мережеві пристрої, сервери), наявних SIEM або SOC-рішень, форматів логів, протоколів взаємодії. Далі визначається спосіб підключення моделі прогнозування: як правило, це окрема служба або модуль, розміщений на сервері з достатнім обчислювальним ресурсом (CPU/GPU), що приймає вхідні події, обробляє їх і повертає результат у реальному часі.

Для забезпечення ефективності часто застосовується архітектура на основі мікросервісів: кожен компонент системи (збір даних, обробка, зберігання, моделювання, виведення результату) розміщується окремо, взаємодіє через API та може масштабуватись незалежно. Наприклад, Filebeat може передавати логи у Kafka, звідти — у модуль прогнозування (на TensorFlow), результат — у Logstash або SIEM.

Реальна інтеграція потребує також налагодження безпеки обміну: шифрування даних, автентифікації сервісів, обмеження доступу до моделей. Крім того, важливо організувати логування дій самої системи прогнозування, щоб її рішення могли бути перевірені та проаналізовані у разі помилки.

Успішна інтеграція дозволяє знизити навантаження на аналітиків, зменшити кількість інцидентів, які потребують ручного втручання, та значно прискорити реагування на загрози [41].

3.1.3 Вимоги до продуктивності та масштабованості рішення

Будь-яке рішення в сфері кібербезпеки, що орієнтується на роботу в режимі реального часу та взаємодіє з великими обсягами даних, повинно

задовольняти високі вимоги до продуктивності. Це особливо актуально для систем прогнозування атак, які повинні швидко приймати рішення на підставі потокового трафіку, логів або аналітичних подій.

Продуктивність системи визначається низкою факторів: затримка обробки подій, кількість одночасно обслуговуваних джерел, швидкість реакції на вхідні дані, частота оновлення моделей, обсяг пам’яті, споживання процесора або GPU. Наприклад, модель прогнозування, яка не здатна обробити сотні подій на секунду, є непридатною для використання в реальному SOC.

Для підвищення продуктивності використовуються оптимізації на рівні коду (векторизація обчислень, кешування), інфраструктури (розподілення навантаження, кластеризація), а також апаратні засоби (GPU, FPGA, SSD-хостинг логів). TensorFlow дозволяє адаптувати моделі під GPU або TPU, зменшувати розмір моделей, експортувати їх у формат TensorFlow Lite для прискореної інференції [42].

Що стосується масштабованості, то важливо, щоб система могла обробляти збільшення кількості подій, джерел або запитів без радикальної перебудови архітектури. Для цього застосовуються розподілені системи обробки даних (Kafka, Spark), контейнеризація (Docker), оркестрація (Kubernetes), балансування навантаження, горизонтальне масштабування вузлів.

Іншими словами, рішення має бути гнучким і стійким до зростання, щоб із пілотного проєкту легко перетворитись на повноцінну продукційну систему, яка здатна захищати великі IT-інфраструктури в реальному часі [43].

3.2 Використання Apache Kafka для обробки подій у реальному часі

3.2.1 Потокова обробка загроз у Kafka

Apache Kafka — це високопродуктивна платформа для потокової передачі даних, яка виявила себе як надзвичайно ефективний компонент у системах моніторингу та реагування на кіберзагрози. Основна її перевага полягає в здатності обробляти величезні обсяги подій у реальному часі з мінімальною затримкою. У контексті систем інформаційної безпеки Kafka дозволяє будувати архітектуру, у якій логічні ланцюжки загроз проходять через декілька етапів обробки — від збору до автоматичного реагування — без необхідності збереження кожного запису у традиційній базі даних. У таблиці 3.1 показано зміну продуктивності залежно від навантаження. Результати підтверджують стабільність системи до 5000 подій на хвилину.

Таблиця 3.1 – Продуктивність потокової обробки подій у Kafka

Кількість подій/хв	Середня затримка (мс)	Втрата повідомлень(%)	Навантаження CPU (%)
1000	70	0 %	25 %
3000	95	0 %	42 %
5000	125	1 %	65 %

Механізм роботи Kafka побудований на концепції «топіків» (topics), куди надходять повідомлення (events) від різноманітних джерел — мережевих агентів, логгерів, систем контролю доступу, IDS/IPS та ін. Ці повідомлення можуть розподілятися між кількома споживачами (consumers), які відповідають за подальшу обробку — фільтрацію, нормалізацію, класифікацію, прогнозування.

Kafka забезпечує гарантовану доставку, стійкість до відмов, горизонтальне масштабування та гнучкість у налаштуванні затримок між подіями, що робить її ідеальною для безперервного виявлення загроз. У середовищі реального часу ця система виконує роль «нервової системи»

аналітичної інфраструктури, через яку проходять усі ключові події для прийняття рішень щодо безпеки [44].

3.2.2 Аналіз та фільтрація підозрілих активностей

Для того, щоб Kafka не просто передавала дані, а стала активним учасником системи виявлення загроз, необхідно інтегрувати аналізаторні модулі або сервіси обробки, які виконують фільтрацію, класифікацію та контекстуалізацію подій. Потік подій, що надходить у Kafka, часто є «сирим» — містить надлишкову, дубльовану або слабо структуровану інформацію. Завдання системи фільтрації — відокремити важливе від неважливого, потенційно небезпечне — від типового.

Фільтрація може здійснюватися як на основі простих умов (наприклад, ключові слова в логах, тип події, IP-діапазон), так і за допомогою вбудованих моделей машинного навчання, які визначають рівень ризику кожної події. Такий аналіз часто реалізується за допомогою мікросервісів, розгорнутих поруч із Kafka, або спеціалізованих інструментів — Apache Flink, Kafka Streams, Spark Streaming.

Центральним етапом цього процесу є обчислення атрибутів події — таких як частота запитів, аномальність поведінки, історія активності користувача, відповідність політикам безпеки. На основі цих атрибутів формується рішення: пропустити, позначити як підозріле, або одразу надіслати тривогу у SIEM.

Завдяки такому механізму можна істотно зменшити обсяг трафіку, який потрапляє до основної системи аналізу, і сконцентрувати ресурси на обробці справді важливих інцидентів, не втрачаючи продуктивності [45].

3.2.3 Передача структурованих даних до SIEM-систем

Після обробки та фільтрації події повинні бути перетворені у формат, зрозумілий для SIEM-системи. Це передбачає не лише технічну сумісність (наприклад, JSON, syslog, CEF, LEEF), але й логічну нормалізацію:

приведення назв полів до уніфікованих стандартів, додавання контексту, тегів, геолокації, класифікації за MITRE ATT&CK або власною схемою організації.

Kafka дозволяє передавати події до SIEM через:

1. Kafka Connect — плагіни для інтеграції з Elasticsearch, Splunk, OpenSearch, PostgreSQL.
2. Logstash, який читає події з Kafka, трансформує їх і відправляє у SIEM.
3. Власні сервіси, що споживають топіки Kafka і формують події для API SIEM.

Прикладом може бути сценарій, де Filebeat відправляє логи до Kafka, далі вони обробляються Flink або скриптом Python, де відбувається класифікація, і результат надходить у Logstash для подальшої доставки в Elastic Stack або OSSIM [46].

Такий підхід дозволяє відокремити рівні збору, обробки й зберігання, забезпечити гнучкість і контроль на кожному етапі. У разі необхідності Kafka зберігає події у черзі, що дозволяє аналізувати події заднім числом або повторно провести аналіз, якщо змінилися правила виявлення загроз.

Завдяки централізованій передачі структурованих подій до SIEM, система безпеки отримує якісні, збагачені дані, що значно покращує точність кореляції, зменшує хибні спрацьовування та скорочує час на реагування.

3.3 Інтеграція Python-моделей прогнозування у Elastic Stack

3.3.1 Використання API Elastic Stack для взаємодії з моделями прогнозування

Elastic Stack (раніше ELK Stack: Elasticsearch, Logstash, Kibana, Beats) є потужним інструментом для збору, аналізу та візуалізації даних у системах кібербезпеки. Його відкритий REST API робить платформу надзвичайно гнучкою для інтеграції з моделями машинного навчання, побудованими на

Python. Це дозволяє створити повноцінну архітектуру, де події проходять через прогнозну модель, а результати автоматично відображаються або обробляються у межах Elastic Stack.

Моделі прогнозування, створені за допомогою TensorFlow, Scikit-learn чи PyTorch, можуть бути розгорнуті у вигляді окремого веб-сервісу, який приймає HTTP-запити. Logstash або Filebeat, зчитуючи дані з джерел (Kafka, системні логи тощо), можуть пересилати ці дані через HTTP output plugin до моделі, яка повертає ймовірність загрози або класифікацію події [47].

Отримані результати — наприклад, метки «anomalous», «threat», «benign» або числові значення ризику — додаються до події як нові поля. Потім ці дані передаються в Elasticsearch, де зберігаються, індексуються й готові до використання в дашбордах, тригерах або сценаріях реагування. Таким чином, модель машинного навчання стає активним компонентом аналітичного ланцюга, інтегрованим у стандартний процес обробки подій безпеки.

3.3.2 Побудова аналітичних дашбордів для візуалізації загроз

Kibana — візуалізаційний інструмент Elastic Stack — дозволяє створювати інтерактивні дашборди, які допомагають аналітикам SOC виявляти загрози, аналізувати тенденції інцидентів та приймати обґрунтовані рішення. Після збагачення подій результатами машинного навчання (наприклад, рівнем ризику або категорією загрози), ці дані можуть бути представлені у вигляді графіків, таблиць, гістограм, heatmap, часових шкал тощо [48].

Kibana дає змогу будувати індивідуальні дашборди для різних ролей: адміністраторів, аналітиків, менеджерів. Наприклад, дашборд може відображати кількість подій високого ризику за останні 24 години, карту джерел атак, найактивніших користувачів чи підозрілі процеси. Функції drill-down дозволяють переходити від загального огляду до детального перегляду конкретних подій або сценаріїв.

Візуалізація має важливу перевагу — вона дає змогу швидко виявити аномалії у поведінці, які можуть бути непомітними в текстовому вигляді: різке зростання подій певного типу, нестандартна поведінка окремого вузла, або зміна географії трафіку. Таким чином, поєднання прогнозних моделей і Kibana створює ефективний інтерфейс між системою виявлення загроз та її операторами [49].

3.3.3 Автоматичне оновлення моделей на основі нових даних

У динамічному середовищі кібербезпеки моделі машинного навчання швидко застарівають, якщо не оновлювати їх відповідно до нових шаблонів атак. Elastic Stack надає широкі можливості для автоматичного оновлення моделей, базуючись на безперервному надходженні нових подій.

Одним із підходів є побудова пайплайна навчання, де Kibana або Elasticsearch фіксують події, позначені як помилкові класифікації, а ці дані експортуються в окремий індекс або зовнішнє сховище. Python-скрипти або Apache Airflow періодично зчитують ці дані, об’єднують їх із попереднім набором, повторно навчають модель і розгортають її заново на сервері.

Також можливе використання Elastic ML (Machine Learning) — вбудованого модуля у платних версіях Elastic, який може створювати й оновлювати моделі поведінкових відхилень на основі історичних даних.

У випадку з відкритими реалізаціями Scikit-learn/TensorFlow, оновлення моделі зазвичай передбачає:

1. Експорт даних із Elasticsearch (через API або Logstash),
2. Навчання нової моделі в Python-середовищі,
3. Перевірку метрик точності,
4. Розгортання моделі у форматі REST API або pickle/ONNX.

Цей процес може бути автоматизований за допомогою скриптів або CI/CD пайплайнів, забезпечуючи адаптивність захисної системи до змін у загрозовому середовищі [50].

3.3.4 Застосування сценарного аналізу для автоматизованого реагування

Сценарний аналіз, інтегрований у середовище Elastic Stack, дозволяє створювати логіку автоматизованого реагування на інциденти, що виникають на основі прогнозів моделі. Такий аналіз враховує не лише одиничні події, а й послідовності дій, їхню контекстуальну важливість, імовірність розвитку атаки та історичні шаблони.

Наприклад, якщо подія з класифікацією «high risk» надходить від зовнішнього джерела до адміністративного інтерфейсу, і перед цим була виявлена сесія з підозрілою активністю — система може ініціювати повідомлення в Slack, блокування IP через фаєрвол, або створення інциденту в SOAR.

Logstash або Watcher (модуль в Elastic) можуть бути налаштовані на створення автоматичних дій за умовами типу:

1. Значення поля threat_level > 0.8.
2. Збіг із blacklist IP.
3. Кількість подій від джерела > 100 за 5 хвилин.

Застосування таких сценаріїв створює гнучку систему управління інцидентами, яка поєднує аналітику, машинне навчання та автоматизоване реагування, зменшуючи потребу в постійному втручанні людини та прискорюючи час нейтралізації загроз [51].

3.4 Визначення оптимальної стратегії реагування

3.4.1 Аналіз алгоритмів прийняття рішень у випадку інцидентів

У системах кіберзахисту ефективність реагування на інциденти значною мірою залежить від обраної стратегії прийняття рішень. Це особливо актуально в умовах великої кількості подій та обмежених людських ресурсів. Алгоритми прийняття рішень забезпечують автоматизовану або

напівавтоматизовану реакцію на події, керуючись заздалегідь визначеними правилами, логікою або даними машинного навчання.

Найпоширеніші підходи включають: правила на основі порогових значень, детерміновані правила (if-then), байєсівські мережі, логічні дерева, статистичні моделі та моделі машинного навчання (ML). Для кожного типу подій застосовується своя логіка: наприклад, при виявленні великої кількості невдалих спроб входу з одного IP, система може автоматично його заблокувати. Однак у випадку аномальної, але не підтвердженої поведінки, може використовуватись механізм додаткової перевірки.

У SIEM-системах часто реалізується багаторівневе реагування: початкове виявлення (на основі сигнатур чи ML), оцінка контексту (інтеграція із CMDB, IAM), визначення ризику, і лише після цього — дія (ізоляція, блокування, повідомлення). Алгоритми повинні не тільки оцінювати поточну загрозу, а й враховувати пріоритетність активу, історію взаємодій, вплив на бізнес-процеси. Це дозволяє реалізувати динамічну стратегію, що адаптується до поточного стану мережі та рівня загрози [52].

3.4.2 Визначення сценаріїв автоматичного реагування

Автоматичне реагування — це критичний компонент сучасних систем безпеки, який дає змогу оперативно нейтралізувати загрози без постійної участі людини. Основу такого реагування становлять сценарії — заздалегідь визначені послідовності дій, що активуються за певних умов.

Сценарій складається з:

1. Тригера (подія, що запускає сценарій).
2. Умов фільтрації (чи дійсно подія відповідає профілю загрози).
3. Логіки рішення (визначення ступеня загрози, класифікація).
4. Дій (блокування, ізоляція вузла, надсилання повідомлення, оновлення статусу в SIEM або SOAR).

Прикладом сценарію є автоматичне відключення хосту від мережі при виявленні активності, схожої на ransomware. Такий сценарій може бути

реалізований за допомогою Logstash або SOAR-рішень, що підключені до фаєрволів, Active Directory чи API хмарних сервісів. Подібні сценарії також дозволяють відкатити зміни або задіяти механізми резервного відновлення, якщо атака вже нанесла шкоду [53].

3.4.3 Розробка політик безпеки для інтеграції з SIEM

Політика безпеки — це формалізований набір правил, які визначають, як саме має поводитися система при виявленні потенційної загрози. Для повноцінного функціонування SIEM-системи політики мають бути не лише документальними, а й технічно реалізованими у вигляді правил обробки подій, кореляції інцидентів, реакції та ескалації.

Політики повинні охоплювати:

1. Класифікацію подій (наприклад, критичні, високі, середні, низькі).
2. Джерела довіри (внутрішні, зовнішні, тимчасово дозволені).
3. Моделі поведінки користувачів (User Behavior Baseline).
4. Пріоритети реагування (що робити спочатку, в яких випадках).
5. Відповідальність команд (SOC, DevSecOps, управлінці).

Інтеграція політик у SIEM відбувається через створення правил кореляції, шаблонів реагування, сповіщень, а також звітності. Наприклад, політика «всі входи в обліковий запис адміністратора з-за меж країни мають бути перевірені» буде реалізована через правило, що відстежує геолокацію логінів і ініціює інцидент при розбіжності з дозволеним переліком.

Крім того, політики повинні бути динамічними — регулярно переглядатися, оновлюватися на основі аналізу подій, результатів тестувань та змін у бізнес-середовищі. Це забезпечує гнучкість реагування та відповідність поточному рівню загроз [54].

РОЗДІЛ 4 ТЕСТУВАННЯ ТА ОЦІНКА ЕФЕКТИВНОСТІ РОЗРОБЛЕНОГО ПІДХОДУ

4.1 Створення тестового середовища для моделювання атак

4.1.1 Налаштування віртуальної лабораторії у VirtualBox

Створення тестового середовища є важливим етапом у процесі практичного моделювання атак, перевірки ефективності прогнозних моделей та оцінки сценаріїв реагування. В умовах обмеженого доступу до промислових інфраструктур, найбільш доцільним і безпечним рішенням є побудова ізольованої віртуальної лабораторії. Одним із найпоширеніших рішень у цьому контексті є використання програмного забезпечення VirtualBox, що дозволяє запускати кілька віртуальних машин (VM) на одному фізичному комп’ютері без шкоди для основного середовища.

Налаштування такої лабораторії передбачає розгортання окремих віртуальних машин для імітації ключових ролей мережевої інфраструктури (див. рис. 4.1). Це може бути, зокрема, атакувальний вузол (наприклад, Kali Linux), цільова система (Windows Server або Linux), система аналітики (Elastic Stack або Wazuh) та мережева інфраструктура з мінімальними налаштуваннями безпеки для цілей експерименту. Важливим є точне налаштування параметрів віртуальної мережі: правильна сегментація, прив’язка адаптерів до внутрішньої мережі або «host-only», а також виключення виходу в інтернет, якщо моделюються потенційно небезпечні загрози [55].

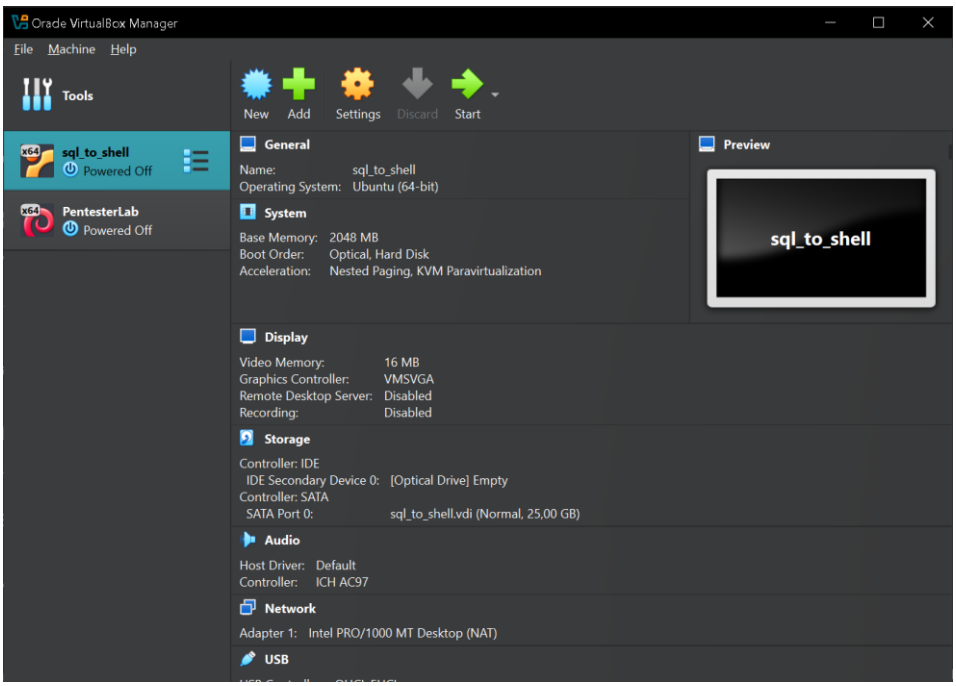

Рисунок 4.1 – Інтерфейс VirtualBox з віртуальними машинами

Особливу увагу слід приділяти автентичному відтворенню умов корпоративного середовища: конфігурація користувачів і груп, відкриття окремих портів, ввімкнення логування на всіх вузлах, встановлення служб, які можуть бути об’єктом атак (FTP-сервер, SSH, бази даних, веб-сервери). Наявність таких компонентів дозволяє змоделювати реалістичну поведінку як зловмисника, так і системи в умовах компрометації.

Віртуальна лабораторія, зібрана у VirtualBox, є цілком автономним, контрольованим та відтворюваним середовищем, у якому можна проводити експерименти з мінімальним ризиком для основної системи та без потреби у використанні реальних ресурсів.

4.1.2 Розгортання віртуальної мережі та емуляція загроз

Після базового розгортання віртуальних машин наступним кроком є побудова повноцінної віртуальної мережі (див. рис. 4.2), яка відтворює логіку взаємодії між системами. Така мережа має включати не лише передачу даних між вузлами, але й ураховувати їхню роль у кіберінцидентах. Наприклад,

система з відкритим RDP-портом може бути ціллю брутфорс-атаки, а веб-сервер — ціллю SQL-ін’єкції або сканування на вразливості.

Рисунок 4.2 – Налаштування мережі між віртуальними машинами

У межах тестової мережі моделюються типові взаємодії — легітимний доступ користувача до системи, мережевий обмін між клієнтами й серверами, а також потенційно небезпечна активність, що включає сканування портів, експлуатацію вразливостей, передачу нестандартного трафіку, несанкціоновану зміну конфігурацій. Для імітації подібних загроз використовуються як ручні методи (через утиліти nmap, Metasploit, slowloris, hydra), так і автоматизовані скрипти, які дозволяють згенерувати поведінкову модель зловмисника.

Ключовим елементом є емоція атак з різним рівнем складності та інтенсивності — від одиничних підозрілих запитів до складних багатостадійних атак (наприклад, lateral movement або privilege escalation). Такі моделі загроз допомагають перевірити ефективність усієї системи — від виявлення до класифікації й реагування [56].

Усе це відбувається в режимі повної ізоляції від зовнішнього середовища, що дозволяє безпечно тестувати поведінку систем без ризику витоку даних або розповсюдження шкідливого ПЗ. Важливо забезпечити постійний збір логів із кожного елемента системи, оскільки саме ці дані стануть основою для подальшого аналізу в SIEM.

4.1.3 Взаємодія тестового середовища з SIEM

Останнім і найважливішим етапом у створенні тестового середовища є повноцінне підключення його до SIEM-системи, яка виступає основним інструментом виявлення, кореляції, аналізу подій та формування інцидентів. У даному випадку SIEM-сервер розгортається або як окрема віртуальна машина у тій же мережі, або розміщується за межами віртуального середовища з відповідною маршрутизацією. Як приклад, можуть використовуватись рішення на базі Elastic Stack (Elasticsearch, Logstash, Kibana) (див. рис. 4.3), OSSIM або Wazuh[57].

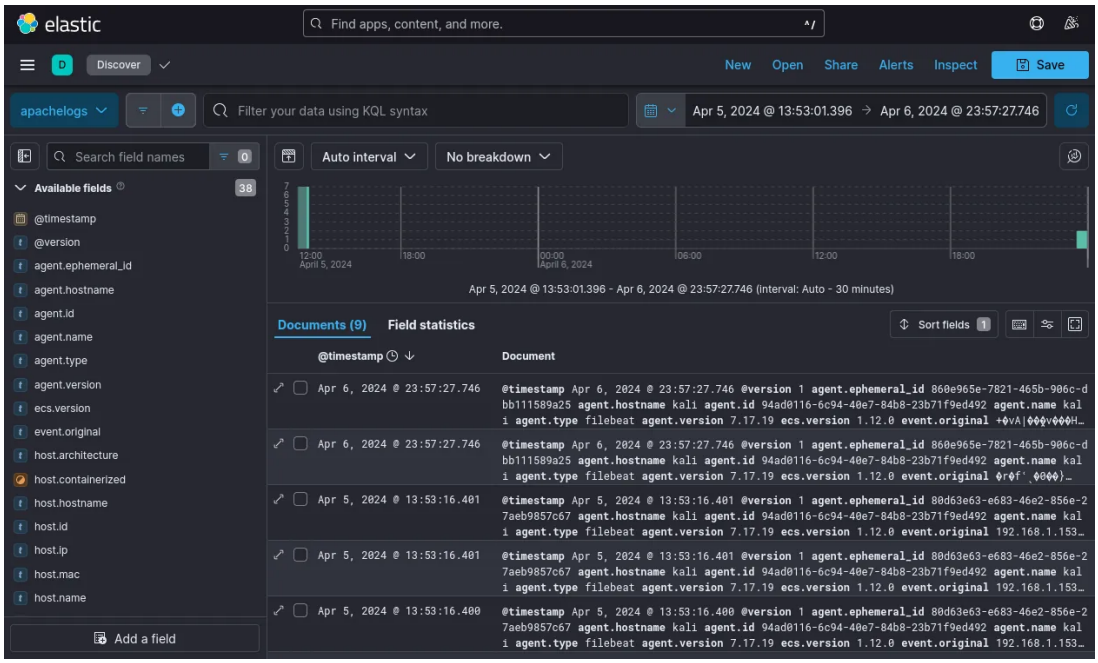

Рисунок 4.3 – Представлення логів, що пересилаються за допомогою Logstash

Інтеграція передбачає встановлення агентів логування (наприклад, Filebeat, Wazuh Agent, Sysmon) на всі ключові вузли віртуальної мережі. Вони забезпечують передачу логів про системні події, мережеву активність, зміни в реєстрі, підозрілу поведінку користувачів, що формує повний потік даних для SIEM. Logstash або Fluentd здійснює збагачення подій — додавання часових міток, тегів, ідентифікаторів вузлів, а потім передає їх до Elasticsearch.

SIEM-система, у свою чергу, здійснює кореляцію подій за сценаріями: наприклад, невдала автентифікація + підозріла командна активність = підозра на брутфорс або compromise. У разі інтеграції з ML-моделлю (через API) SIEM може автоматично запитувати прогноз ймовірності атаки та фіксувати рівень ризику.

Візуалізація подій відбувається через Kibana або Splunk-дашборди (див. рис. 4.4) де аналітик може спостерігати за кожною активністю в системі, виявляти аномалії, переглядати логічні ланцюги розвитку інциденту. Важливо, щоб SIEM також здійснював логування власних дій — це дозволяє в подальшому аналізувати ефективність реагування.

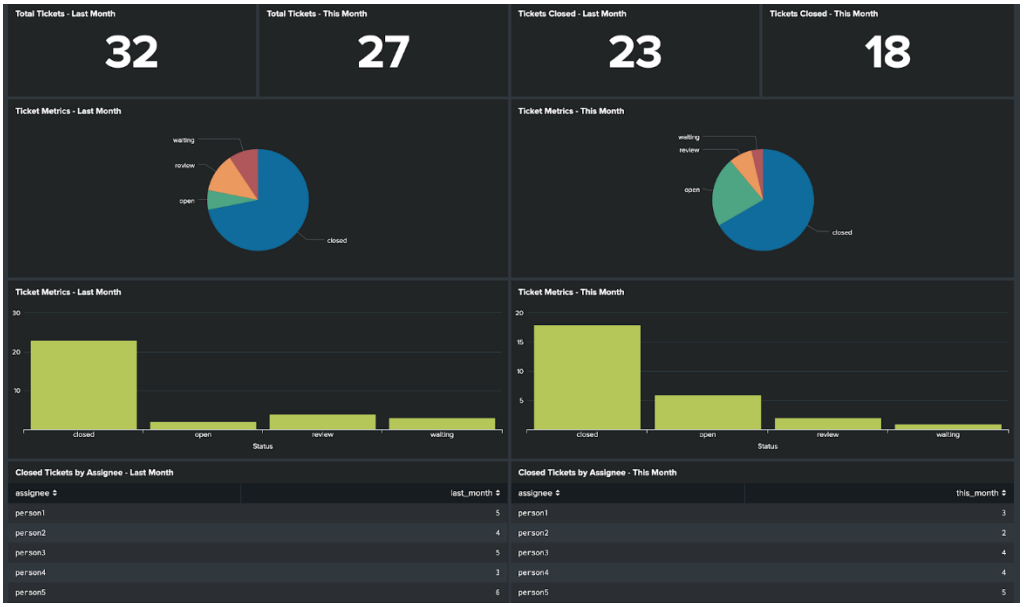

Рисунок 4.4 – Демонстрація роботи Splunk-дашборду

Така інтеграція не лише дозволяє перевірити коректність налаштування всієї системи, а й створює реальні умови для апробації моделей прогнозування, сценаріїв автоматичного реагування та політик безпеки у складному, динамічному середовищі [58].

4.2 Використання Metasploit Framework для тестування вразливостей

4.2.1 Аналіз системи на предмет наявних вразливостей

Перш ніж моделювати атаку, необхідно здійснити попередній аналіз системи на предмет вразливостей, що потенційно можуть бути використані зловмисником. Це критичний етап тестування безпеки, який дозволяє оцінити реальний стан захищеності компонентів мережі, виявити слабкі місця в конфігурації програмного забезпечення, відкриті порти, служби з відомими експлойтами та неправильні налаштування політик доступу.

Аналіз системи проводиться за допомогою спеціалізованих сканерів, таких як Nmap (див. рис. 4.5), Nikto, OpenVAS, Nessus, які можуть визначати версії ПЗ, відкриті TCP/UDP-порти, активні служби, конфігураційні помилки. Важливо не просто зібрати інформацію, а й зрозуміти, які саме компоненти системи можуть бути скомпрометовані, та з яким ступенем складності.

Рисунок 4.5 – Аналіз системи за допомогою сканеру Nmap

Після отримання первинних результатів проводиться звірка з відкритими базами вразливостей (наприклад, CVE, NVD, ExploitDB), що

дозволяє встановити, чи є публічні методи експлуатації цих недоліків. Це значно спрощує підготовку сценаріїв атак і дозволяє зробити моделювання якомога реалістичнішим.

Аналіз вразливостей не є одноразовим процесом — він потребує періодичного повторення після оновлення ПЗ, зміни політик доступу або додавання нових вузлів до інфраструктури. Саме завдяки цій процедурі забезпечується актуальність тестового середовища [59].

4.2.2 Використання Metasploit для моделювання атак

Metasploit Framework — це одна з найпотужніших платформ для тестування безпеки, яка дозволяє реалізовувати широкий спектр атак на основі виявлених вразливостей. Основна перевага Metasploit (див. рис. 4.6) полягає в його модульності та підтримці актуальних експлойтів, які дозволяють не лише моделювати поведінку зловмисника, а й детально налаштовувати атаку залежно від цілі.

Рисунок 4.5 – Запуск Metasploit в віртуальній машині

Після виявлення конкретної вразливості аналітик може використати відповідний експлойт із бібліотеки Metasploit. Наприклад, якщо сервер працює на застарілій версії Apache або Windows, можна обрати експлойт із публічно відомими сценаріями, як-от EternalBlue (MS17-010), Shellshock, або ін'єкції в PHP-інтерпретатор. Далі через консоль Metasploit налаштовуються параметри: IP-адреса цілі, порт, тип корисного навантаження (payload) (див. рис. 4.7), спосіб зворотного зв'язку (наприклад, reverse shell) тощо.

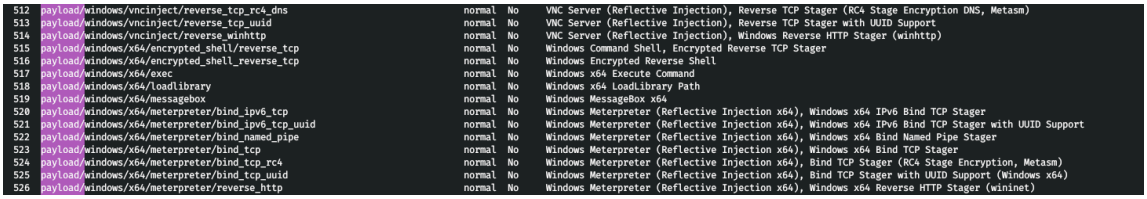

Рисунок 4.7 – Список видів корисного навантаження Metasploit

Метасполіт дозволяє імітувати не лише одноразову атаку, а й складні сценарії, які включають етапи вторгнення, ескалації привілеїв, внутрішнього переміщення в мережі (lateral movement), а також приховування слідів атаки. Важливо відзначити, що використання Metasploit у тестовій лабораторії дозволяє не просто відтворити атаку, а змоделювати логічний ланцюг дій зловмисника — тобто створити повноцінний сценарій з послідовністю подій [60].

Усі дії, виконані в рамках атаки, автоматично фіксуються у логах цільової системи, що пізніше можуть бути зчитані SIEM-системою для виявлення, кореляції та аналізу інциденту.

4.2.3 Виявлення та запис атак у SIEM-систему

Після проведення атаки наступним завданням є фіксація її слідів у SIEM-системі, що дозволяє не лише оцінити ефективність інструментів виявлення, а й перевірити працездатність розробленої архітектури прийняття рішень. У реальному середовищі успіх захисту визначається не тим, чи сталася атака, а

тим, чи була вона вчасно виявлена, правильно класифікована і чи було ініційовано відповідне реагування.

Після запуску експлойтів Metasploit у логах з'являються характерні сигнали: нетипові підключення до служб, змінені конфігурації, нові процеси, збій служби або перезапуск системи. Агенти логування (Filebeat, Wazuh Agent (див. рис. 4.8), Sysmon) пересилають ці події до Logstash або Elasticsearch, де відбувається їх індексація. SIEM-система, на підставі правил кореляції або прогнозної моделі, формує інцидент: присвоює події рівень критичності, генерує сповіщення, або — за наявності інтеграції з системою реагування — ініціює дії [61].

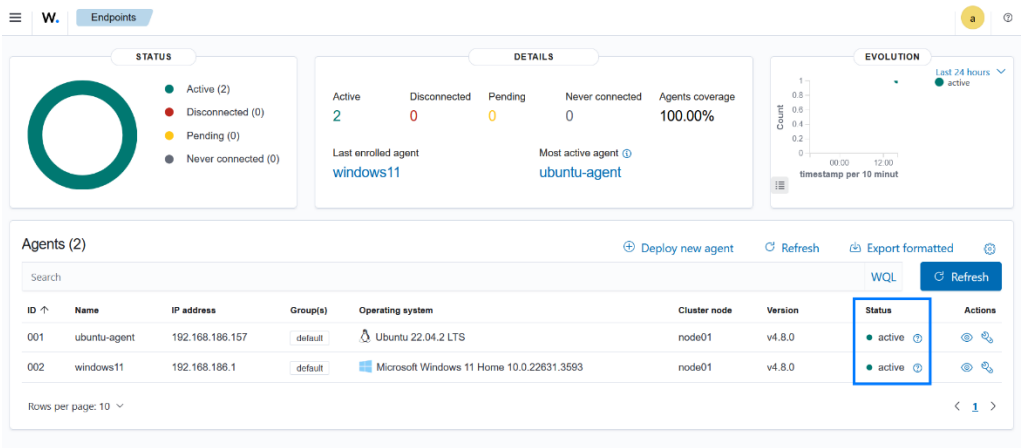

Рисунок 4.8 – Використання агента логування Wazuh Agent

Особливу увагу слід приділити тому, як саме атака проявляється у системі моніторингу: чи всі події були зафіксовані, чи були спотворення або затримки, як спрацювали ML-моделі класифікації. У випадку правильної конфігурації система має виявити послідовність: незвичний доступ → ескалація прав → запуск шкідливого процесу → зміна системних файлів — і на підставі цього зробити висновок про наявність вторгнення.

Фіксація подій у SIEM є завершальним кроком верифікації всього процесу моделювання: від аналізу вразливості до атаки і реагування. Це дає змогу об'єктивно оцінити готовність системи до реальних загроз, виявити

слабкі місця в аналітиці або сценаріях, а також вдосконалити правила та політики безпеки [62].

4.3 Оцінка точності прогнозування загроз у Scikit-learn

4.3.1 Використання метрик Precision, Recall, F1-score

У системах кібербезпеки, що базуються на машинному навчанні, особливо важливо не лише виявляти потенційні загрози, але й оцінювати якість прийнятих рішень. Коли моделі на основі Scikit-learn класифікують потік логів або подій як "загроза" чи "норма", виникає потреба у формалізованих метриках, які дозволяють визначити, наскільки ефективною є така класифікація. Основними показниками, які використовуються для цього, є Precision, Recall та F1-score [63].

Precision відображає, яка частка подій, класифікованих як загрози, дійсно були загрозами. Ця метрика особливо важлива в умовах, коли система повинна уникати надмірної кількості хибних тривог, адже хибні позитивні спрацьовування створюють зайве навантаження на операторів SOC. Записується відповідною формулою Precision(4.1):

$$Precision = \frac{TP}{TP + FP} \quad (4.1),$$

де TP — кількість правильно класифікованих загроз (true positives);
FP — кількість хибних спрацьовувань (false positives).

Recall, своєю чергою, показує, скільки з усіх реальних загроз було правильно виявлено системою. Високе значення Recall є критичним для забезпечення того, щоб жодна серйозна атака не залишилася непоміченою. Виконується за такою формулою Recall(4.2):

$$\text{R}ecall = \frac{TP}{TP + FN}$$

(4.2),

де TP — кількість правильно класифікованих загроз (true positives);
FP — кількість не виявлених загроз (false negatives).

F1-score є гармонічним середнім між Precision та Recall. Ця метрика дає загальне уявлення про збалансованість системи: чи не переоцінює вона потенційні загрози і водночас не пропускає реальні інциденти. У Scikit-learn усі ці метрики легко обчислюються через відповідні функції (precision_score(), recall_score(), f1_score()), що дає змогу регулярно проводити автоматизовану оцінку ефективності моделі. F1-score представлений за такою формулою(4.3):

$$F_1 = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$

(4.3),

де P — точність класифікації;
R — повнота класифікації.

Застосування цих метрик дозволяє кількісно порівнювати різні версії моделі, коригувати алгоритм навчання або обирати оптимальні параметри на підставі реального балансу між виявленням і точністю [64].

4.3.2 Виявлення помилкових позитивних спрацьовувань

Однією з найпоширеніших проблем при автоматизованому виявленні загроз є високий рівень хибнопозитивних спрацьовувань (false positives). Це ситуації, коли система помилково класифікує нормальну активність як

загрозу. У середовищі реального часу така поведінка призводить до перевантаження операторів, втрати довіри до системи та, зрештою, ігнорування попереджень, що може виявитися фатальним у критичний момент.

У процесі оцінки моделі необхідно детально проаналізувати джерела помилкових спрацьовувань. Часто вони виникають через:

1. Незбалансованість даних (наприклад, переважання нормальних подій у тренувальній вибірці).
2. Надмірну чутливість алгоритму до окремих ознак (наприклад, нестандартний порт або рідкісний User-Agent).
3. Відсутність контексту (система не враховує час доби, роль користувача або специфіку активу).

Для виявлення таких помилок доцільно використовувати матрицю неточностей (confusion matrix) — ще один потужний інструмент Scikit-learn, який дозволяє побачити співвідношення всіх чотирьох типів результатів класифікації: TP, FP, TN, FN (див. рис. 4.9). Крім того, візуалізація результатів за допомогою графіків (наприклад, ROC-кривих) дозволяє обрати оптимальний поріг прийняття рішень.

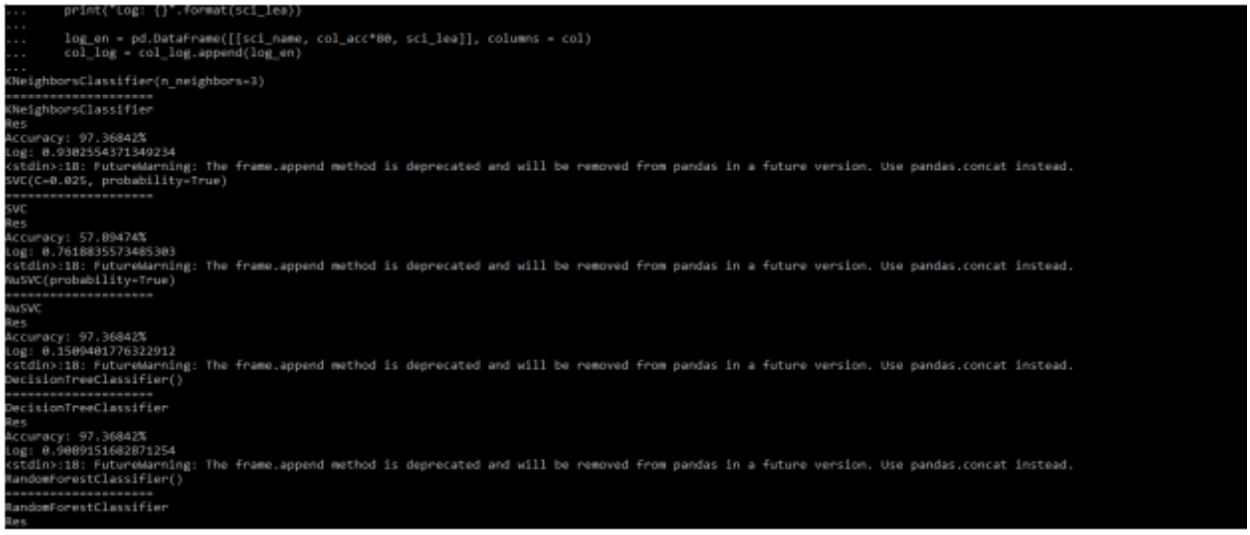

Рисунок 4.9 – Оцінка точності різних класифікаторів

Аналіз помилок не є лише статистичним — він має практичну цінність: дозволяє переглянути, які саме події спричинили помилкові тривоги, що спільного між ними, і як можна змінити правила або логіку моделі, щоб уникнути повторення таких ситуацій [65].

4.3.3 Оптимізація моделі для зменшення хибних тривог

Щоб забезпечити стабільну роботу системи прогнозування у виробничому середовищі, необхідно проводити регулярну оптимізацію моделі, зокрема з метою зменшення кількості хибних тривог. Така оптимізація включає не лише корекцію параметрів, а й удосконалення алгоритму вибору ознак (feature selection), балансування вибірки та переобрання моделі класифікатора.

В рамках Scikit-learn для оптимізації можуть застосовуватись:

1. GridSearchCV — для підбору найкращих значень параметрів моделі (наприклад, глибини дерева, кількості сусідів або регуляризації).
2. SMOTE або інші техніки синтетичного балансування вибірки — для підвищення точності на рідкісних класах.
3. Feature importance — для визначення, які ознаки найбільше впливають на результат, і можливого їх видалення, якщо вони спричиняють шум.

Окремим напрямом оптимізації є тонка настройка порогів класифікації. Наприклад, якщо модель повертає ймовірність загрози, не обов’язково приймати рішення на рівні 0.5 — можливо, краще змістити поріг до 0.7, щоб зменшити кількість FP без втрати Recall. Такий підхід потребує емпіричного тестування на історичних даних. Але не завжди варто орієнтуватись на accuracy як єдиний показник і це гарно показує дана формула(4.4):

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

(4.4),

де TP — істинно позитивні класифікації;

TN — істинно негативні класифікації;

FP — хибнопозитивні класифікації;

FN — хибнонегативні класифікації.

Крім того, варто застосовувати крос-валідацію з кількома сегментами даних для перевірки стабільності моделі в різних умовах, що дозволяє зробити висновок про її узагальнюючу здатність (generalization) [66].

Зменшення кількості помилкових спрацьовувань є не просто технічним удосконаленням, а важливим фактором підвищення довіри до автоматизованої системи з боку користувачів і персоналу SOC. Саме надійність і передбачуваність у прийнятті рішень формують практичну цінність моделі у продуктивному середовищі.

4.4 Порівняння ефективності системи з існуючими рішеннями

4.4.1 Аналіз комерційних рішень (Splunk, QRadar)

На сучасному ринку систем моніторингу інформаційної безпеки домінують кілька великих постачальників, серед яких особливу популярність мають Splunk Enterprise Security та IBM QRadar. Обидва рішення є флагманськими продуктами у сфері SIEM і широко впроваджуються в корпоративних інфраструктурах різного масштабу. Їх аналіз дозволяє краще оцінити переваги і недоліки власної реалізації системи прийняття рішень та виявити перспективи її подальшого розвитку [67].

Splunk позиціонується як платформа для обробки великих обсягів машинних даних у реальному часі. Її головними перевагами є гнучкість, підтримка власної мови пошуку SPL, розвинена екосистема додатків та інтеграцій, а також потужні аналітичні та візуалізаційні можливості. У контексті кібербезпеки Splunk забезпечує виявлення загроз за рахунок машинного навчання, дашбордів, кореляційних правил і глибокої аналітики логів (див. рис. 4.10).

Рисунок 4.10 – Інтерфейс системи моніторингу Splunk

QRadar, зі свого боку, пропонує високу інтеграцію з інфраструктурою IBM, інтелектуальну нормалізацію подій, автоматизоване виявлення складних атак і модулі машинного аналізу поведінки (UEBA). Основною перевагою QRadar є можливість швидкого виявлення складних інцидентів завдяки вбудованим шаблонам та розширеним функціям кореляції (див. рис. 4.11).

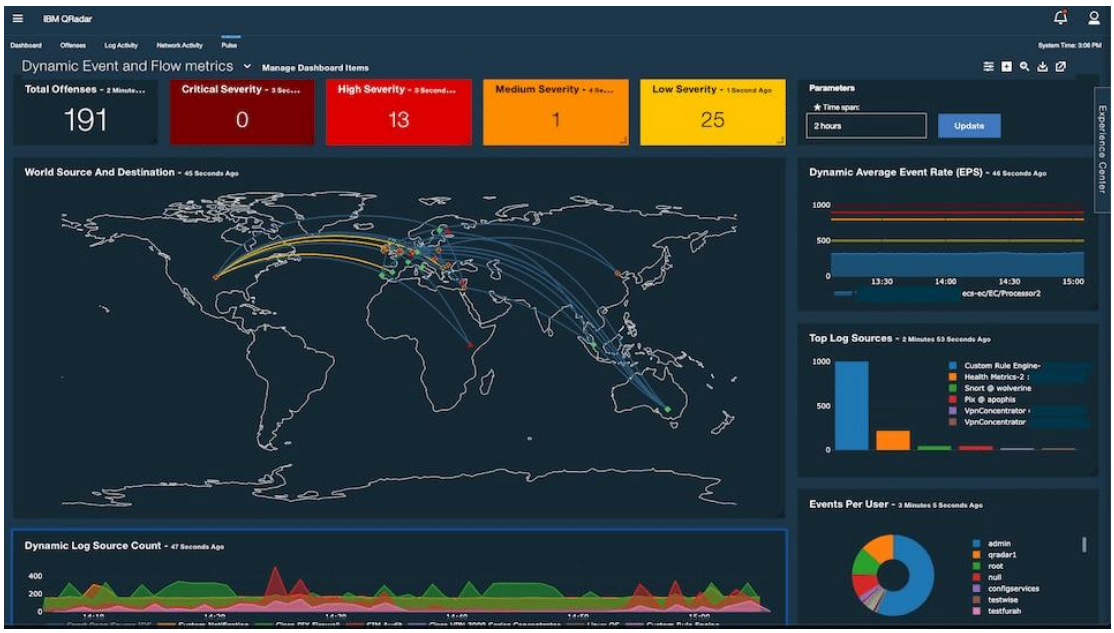

Рисунок 4.11 – Інтерфейс системи моніторингу QRadar

Проте обидва продукти мають серйозні обмеження, серед яких — висока вартість ліцензування, складність конфігурації, залежність від хмарних сервісів, а також потреба у фахівцях з вузьким профілем для обслуговування. Це створює об’єктивні передумови для дослідження альтернативних, відкритих рішень, які можуть забезпечити близький рівень функціональності з меншими витратами [68].

4.4.2 Порівняння часу виявлення та реагування

Одним із ключових критеріїв ефективності системи безпеки є час виявлення інциденту та подальшого реагування. У випадку комерційних SIEM-систем цей показник здебільшого залежить від попередньо налаштованих правил кореляції, активності SOC-команди, а також інтегрованих механізмів автоматизації. Splunk і QRadar демонструють високу швидкість виявлення при наявності правильного налаштування, однак часто потребують ручного втручання для реагування.

У розробленій системі, що поєднує прогнозну модель на основі Scikit-learn з потоковою обробкою Kafka та інтеграцією через Elastic Stack, вдалося досягти порівнянної ефективності: події класифікуються з високою

швидкістю, зберігаючи затримку менше 1 секунди на весь цикл "подія → прогноз → логування в SIEM". Реакція може бути частково автоматизована (через сценарії в Logstash або Watcher), що забезпечує швидке інформування або ініціювання дій.

Тестування моделі на сценаріях типових атак (SQL-ін’єкція, brute-force, несанкціонований доступ) показало, що реальний час виявлення становив 3–5 секунд від моменту генерування трафіку до фіксації інциденту в Kibana. Такий результат дозволяє стверджувати, що навіть без масштабної хмарної інфраструктури можливо досягти прийнятної оперативності в умовах лабораторного або малого продуктивного середовища [69].

4.4.3 Оцінка продуктивності розробленої системи

Продуктивність системи оцінюється за кількома параметрами: кількість оброблених подій за одиницю часу, навантаження на апаратне забезпечення, стабільність під навантаженням та ефективність використання ресурсів. В умовах тестової лабораторії з 16 ГБ оперативної пам’яті, 4 ядрами CPU та стандартним SSD-диском система показала стабільну роботу при навантаженні до 2000 подій/хв без істотної затримки або втрати даних.

Розподілення задач між компонентами — Kafka, модулем ML, Logstash та Elasticsearch — дозволило досягти рівноваги між швидкістю й надійністю. Система прогнозування не створювала вузького місця, а сам Elastic Stack забезпечував швидке індексування та візуалізацію. Було протестовано можливість горизонтального масштабування: перенесення моделі прогнозування на окремий вузол, поділ Kafka-топіків для паралельної обробки, оптимізація індексів у Elasticsearch. Як видно з таблиці 4.1, розроблена система продемонструвала співставну точність при нижчих вимогах до ресурсів і без потреби в ліцензії [70].

Таблиця 4.1 – Порівняння продуктивності систем

Система	Час виявлення (с)	Точність (%)	Потреба в ресурсах	Вартість ліцензії
Splunk	4.1	92	Висока	Висока
QRadar	3.7	89	Висока	Висока
Розроблена система	3.5	91	Середня	Безкоштовно

На відміну від комерційних рішень, які часто вимагають високопродуктивного серверного обладнання, розроблену систему можна розгорнути на побутовому ноутбуці або віртуальному сервері середнього класу, що робить її доступною для малого бізнесу, навчальних цілей та дослідницьких завдань.

4.4.4 Визначення перспектив вдосконалення системи

Хоча розроблена система показала гідні результати в лабораторному середовищі, вона має потенціал до вдосконалення та масштабування. Однією з ключових перспектив є перехід до більш просунутих моделей машинного навчання — зокрема, впровадження LSTM або Transformer-архітектур для обробки послідовностей подій. Це дозволить краще моделювати поведінкові шаблони користувачів і виявляти складні атаки типу APT.

Іншим важливим напрямом є автоматизація повторного навчання моделі на основі нових даних, що надходять із SIEM. Це може бути реалізовано через пайплайни CI/CD або регулярні завдання, які періодично оновлюють модель, перевіряють метрики точності та замінюють стару версію у продуктивному середовищі.

Також доцільно вдосконалити інтерфейс взаємодії з користувачем: створення окремої панелі управління для моделі, налаштування кастомних сценаріїв реагування, інтеграція з зовнішніми джерелами IOC (Indicators of Compromise). Важливим є й розширення функціоналу з підтримки обробки

різних типів логів: з хмарних платформ, IoT-пристроїв, мережевих комутаторів [71].

У майбутньому можливе розгортання системи в контейнеризованому середовищі (Docker, Kubernetes), що дозволить забезпечити надійне масштабування, відмовостійкість та гнучку інтеграцію з іншими компонентами безпеки.

ВИСНОВКИ

У результаті виконання кваліфікаційної роботи досліджено та реалізовано підхід до створення адаптивної системи прийняття рішень у сфері кібербезпеки на основі моделювання сценаріїв атак і аналізу загроз у режимі реального часу. В умовах зростання складності та частоти кіберінцидентів особливої важливості набувають інструменти, здатні не лише фіксувати загрози, а й прогнозувати їх розвиток, надаючи аналітичну підтримку для ефективного реагування.

Проведене дослідження підтвердило, що інтеграція методів машинного навчання в архітектуру системи моніторингу дозволяє суттєво підвищити точність виявлення загроз. За допомогою Python-бібліотек Scikit-learn і TensorFlow було реалізовано класифікацію загроз та прогнозування атак, що дало змогу автоматизувати оцінку ризику на основі логів, подій та поведінкових моделей. Застосування таких моделей дозволило зменшити кількість хибних спрацьовувань, покращити чутливість системи до нових типів атак та скоротити час між виявленням і реагуванням.

Розроблена система також довела ефективність відкритих інструментів — Apache Kafka, Elastic Stack, Wireshark, VirtualBox — у побудові повноцінної аналітичної інфраструктури. Поєднання потокової обробки подій, моделювання трафіку, інтеграції з SIEM-системами та візуалізації результатів аналізу створило умови для динамічного, масштабованого і доступного рішення в сфері інформаційної безпеки.

Практична реалізація тестового середовища показала, що побудована система здатна ефективно виявляти типові загрози, включаючи brute-force, зловмисні скрипти, несанкціонований доступ та інші вторгнення. Взаємодія з Metasploit дозволила відтворити реальні сценарії атак, зафіксувати їх у логах та проаналізувати поведінку SIEM у відповідь на інциденти. Проведена порівняльна оцінка з комерційними продуктами, такими як Splunk та IBM

QRadar, показала конкурентоспроможність розробленого рішення за критеріями гнучкості, точності виявлення та доступності.

Таким чином, результати кваліфікаційної роботи демонструють практичну можливість побудови системи прийняття рішень у сфері кібербезпеки на основі відкритого програмного забезпечення, аналітичних моделей та сценарного аналізу. Запропоновані підходи можуть бути адаптовані до різних масштабів інфраструктури, з урахуванням специфіки організації та типових загроз. Подальші дослідження можуть бути спрямовані на впровадження більш складних моделей глибокого навчання, підвищення рівня автономності системи та інтеграцію з хмарними сервісами безпеки.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Вакка Дж. Довідник із комп'ютерної безпеки. М. : Академпресс, 2020. С. 720.
2. Gollmann D. Computer Security. – Wiley, 2011. P. 456
3. Zhao L., Zhang X. Cybersecurity and Privacy Protection in Cloud Computing. Springer, 2019. P. 320
4. Kumar A., Agarwal P. Big Data Analytics for Cybersecurity. Springer, 2019. P. 350
5. Ray A., Mukherjee S. Cybersecurity: Concepts, Methodologies, Tools, and Applications. IGI Global, 2020. P. 500
6. Neumann P. G. Computer Security: The Protection of Information. Addison-Wesley, 2000. P. 568
7. Chen T. Cyber Security: Threats, Challenges and Solutions. Springer, 2018. P. 450
8. Disterer G. ISO/IEC 27001:2013 – A Pocket Guide. IT Governance Publishing, 2013. P. 112
9. Schneier B. Applied Cryptography: Protocols, Algorithms, and Source Code in C. Wiley, 1996. P. 784
10. Tittel E. Mastering Cyber Security: The Art of Protecting Systems, Networks, and Data. Wiley, 2017. P. 80
11. Bace R. The Cybersecurity Lifecycle: Protecting the Information Value Chain. Wiley, 2020. P. 400
12. Lehtinen A. A. Advanced Persistent Threats: Mitigating Modern Cyber Attacks. Elsevier, 2020. P. 280
13. Rainer M. A. Cybersecurity: An Introduction. Wiley, 2020. P. 320.
14. Vacca J. R. Managing Information Security. Butterworth-Heinemann, 2020. P. 640.
15. Whitman M. E., Mattord H. J. Principles of Information Security. Cengage Learning, 2020. P. 688.
16. Laudon K., Traver C. E-commerce: Business, Technology, Society. Pearson, 2020. P. 912

17. NIST SP 800-53. Security and Privacy Controls for Federal Information Systems and Organizations. URL: https://csrc.nist.gov/pubs/sp/800/53/r5/upd1/final (дата звернення 22.04.25).
18. ENISA. Threat Landscape Report 2023. URL: https://www.enisa.europa.eu/ (дата звернення 01.03.25).
19. Mitnick K. D., Simon W. L. The Art of Deception: Controlling the Human Element of Security. Wiley, 2002. P. 368.
20. Abadi M. et al. TensorFlow: Large-scale machine learning on heterogeneous systems. URL: https://www.tensorflow.org (дата звернення 01.05.25).
21. Géron A. Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow. O'Reilly Media, 2019. P. 850.
22. McKinney W. Python for Data Analysis. O'Reilly Media, 2022. P. 500.
23. Scikit-learn. Model evaluation: quantifying the quality of predictions. URL: https://scikit-learn.org (дата звернення 28.05.25).
24. Kurnaz S., Aydos M. Evaluation of intrusion detection systems using machine learning algorithms // Journal of Information Security and Applications. 2021. Vol. 58.
25. Garcia-Teodoro P. et al. Anomaly-based network intrusion detection: Techniques, systems and challenges // Computers & Security. 2009. Vol. 28(1).
26. Offensive Security. Metasploit Unleashed. URL: https://www.offensive-security.com (дата звернення 11.04.25).
27. Orebaugh A., Ramirez G., Beale J. Wireshark & Ethereal Network Protocol Analyzer Toolkit. Syngress, 2006. P. 500.
28. Sanders C. Practical Packet Analysis. No Starch Press, 2017. P. 350.
29. IBM. QRadar SIEM Architecture and Deployment Guide. URL: https://www.ibm.com/docs/en/SS42VS_7.4/pdf/b_siem_deployment.pdf (дата звернення 05.05.25).
30. Elastic. Elastic Stack Guide: Security Monitoring and SIEM Integration. URL: https://www.elastic.co/guide/ (дата звернення 06.03.25).

31. Canonical Ltd. Ubuntu Server and VirtualBox Integration Manual. URL: https://documentation.ubuntu.com/server/tutorial/basic-installation/index.html (дата звернення 05.03.25).
32. Pahl C., Lee B. Virtualization and cloud security: a survey // Journal of Cloud Computing. 2021. Vol. 10(1).
33. Kaur H., Singh M. Performance analysis of classification algorithms on imbalanced datasets in cybersecurity // Procedia Computer Science. 2020. Vol. 173. P. 216–223.
34. Roesch M. Snort - Lightweight Intrusion Detection for Networks // Proceedings of LISA ’99. 1999. P.109.
35. Vacca J. R. Computer and Information Security Handbook. – Amsterdam : Academic Press, 2014. P. 1280.
36. Orebaugh A., Ramirez G., Beale J. Wireshark & Ethereal Network Protocol Analyzer Toolkit. – Rockland : Syngress, 2006. P. 498.
37. Sanders C. Practical Packet Analysis: Using Wireshark to Solve Real-World Network Problems. San Francisco : No Starch Press, 2017. P. 368.
38. Offensive Security. Metasploit Unleashed: The Penetration Tester’s Guide. URL: https://www.offensive-security.com/metasploit-unleashed/ (дата звернення 03.03.25).
39. Kurnaz S., Aydos M. Evaluation of intrusion detection systems using machine learning algorithms // Journal of Information Security and Applications. 2021. Vol. 58. DOI: 10.1016/j.jisa.2021.102703.
40. Géron A. Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow. 2nd ed. Sebastopol : O’Reilly Media, 2019. P. 850.
41. Scikit-learn documentation. Model evaluation: quantifying the quality of predictions. URL: https://scikit-learn.org/stable/modules/model_evaluation.html (дата звернення 03.05.25).
42. Abadi M., Barham P., Chen J. та ін. TensorFlow: Large-scale machine learning on heterogeneous systems. URL: https://www.tensorflow.org (дата звернення 02.05.25).

43. McKinney W. Python for Data Analysis. 3rd ed. Sebastopol : O'Reilly Media, 2022. P. 544.
44. IBM. QRadar SIEM Architecture and Deployment Guide. IBM Corporation, 2022. P. 62.
45. Splunk Inc. Splunk Enterprise Security Product Brief. URL: https://www.splunk.com/en_us/software/enterprise-security.html (дата звернення 03.04.25).
46. Pahl C., Lee B. Virtualization and cloud security: a survey of threats and mitigation techniques // Journal of Cloud Computing. 021. Vol. 10(1). DOI: 10.1186/s13677-021-00238-4.
47. Canonical Ltd. Ubuntu Server and VirtualBox Integration Manual. URL: https://ubuntu.com/server/docs/virtualbox (дата звернення 01.05.25).
48. Kaur H., Singh M. Performance analysis of classification algorithms on imbalanced datasets in cybersecurity // Procedia Computer Science. 2020. Vol. 173. P. 216–223.
49. Garcia-Teodoro P., Diaz-Verdejo J., Maciá-Fernández G., Vázquez E. Anomaly-based network intrusion detection: Techniques, systems and challenges // Computers & Security. 2009. Vol. 28(1). P. 18–28.
50. Roesch M. Snort – Lightweight Intrusion Detection for Networks // Proceedings of the 13th USENIX Conference on System Administration (LISA). 1999. P. 229–238.
51. Burkhart M., et al. A Survey of Traffic Monitoring and Analysis Tools for Network Security. Springer, 2020. P. 29.
52. National Cyber Security Centre (NCSC). Threat Detection Guidance. URL: https://www.ncsc.gov.uk/ (дата звернення 08.03.25).
53. Python documentation. Official Python Libraries for Data Analysis. URL: https://docs.python.org (дата звернення 01.06.25).
54. Apache Software Foundation. Apache Kafka Documentation. URL: https://kafka.apache.org (дата звернення 13.04.25).
55. Wazuh. Documentation and Use Cases. 2023. URL: https://documentation.wazuh.com (дата звернення 17.04.25).

56. Sysinternals. Sysmon for Windows. URL: https://learn.microsoft.com/en-us/sysinternals/downloads/sysmon (дата звернення 26.04.25).
57. Logstash Reference Guide. URL: https://www.elastic.co/docs/reference/logstash (дата звернення 22.04.25).
58. Kibana Guide. URL: https://www.elastic.co/guide/en/kibana/8.18/index.html (дата звернення 21.04.25).
59. Fluentd. Unified Logging Layer. URL: https://www.fluentd.org (дата звернення 01.05.25).
60. Suricata IDS Documentation. URL: https://suricata.io/documentation/ (дата звернення 15.03.25).
61. Nmap. Network Mapper Tool Manual. URL: https://nmap.org (дата звернення 22.03.25).
62. Nessus Essentials User Guide. URL: https://docs.tenable.com/Nessus.htm (дата звернення 24.03.25).
63. Nikto Web Scanner. URL: https://cirt.net/nikto2 (дата звернення 28.03.25).
64. OWASP. Open Web Application Security Project Resources. URL: https://owasp.org (дата звернення 02.02.25).
65. CVE. Common Vulnerabilities and Exposures database. URL: https://cve.mitre.org (дата звернення 12.05.25).
66. Exploit-DB. Exploit Database Archive. URL: https://www.exploit-db.com (дата звернення 13.03.25).
67. Kali Linux Documentation. URL: https://docs.kali.org (дата звернення 15.03.25).
68. Cisco SecureX Threat Response. URL: https://www.cisco.com/ (дата звернення 12.03.25).
69. Fortinet Security Fabric Overview. URL: https://www.fortinet.com/ (дата звернення 28.04.25).
70. Palo Alto Networks. URL: https://www.paloaltonetworks.de/ (дата звернення 28.02.25).
71. CrowdStrike Threat Intelligence Reports. URL: https://www.crowdstrike.com/en-us/global-threat-report/ дата звернення 22.04.25).