

**НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені
ІГОРЯ СІКОРСЬКОГО»
НАВЧАЛЬНО-НАУКОВИЙ ФІЗИКО-ТЕХНІЧНИЙ ІНСТИТУТ
Кафедра інформаційної безпеки**

«На правах рукопису»
УДК _____

«До захисту допущено»
Завідувач кафедри
_____ Дмитро ЛАНДЕ
(підпис)
«_____» _____ 2025 р.

**Магістерська дисертація
на здобуття ступеня магістра
за освітньо-професійною програмою «Системи, технології та
математичні методи кібербезпеки»
спеціальності 125 «Кібербезпека та захист інформації»**

на тему: Модель машинного навчання для виявлення зловмисної активності в
журналах подій.

Виконав (-ла): здобувач вищої освіти ІІ курсу, групи ФБ-41мп
(шифр групи)

Дрига Артем Вадимович
(прізвище, ім'я, по батькові) _____ (підпис)

Керівник к.т.н., доцент Барановський О. М.
(посада, науковий ступінь, вчене звання, прізвище, ім'я, по батькові) _____ (підпис)

Рецензент к.т.н.,
(посада, науковий ступінь, вчене звання, науковий ступінь, прізвище, ім'я, по батькові) (підпис)

Засвідчую, що у цій магістерській дисертації немає
запозичень з праць інших авторів без відповідних
посилань.

Здобувач вищої освіти _____
(підпис)

Київ - 2025 року

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
НАВЧАЛЬНО-НАУКОВИЙ ФІЗИКО-ТЕХНІЧНИЙ ІНСТИТУТ
Кафедра інформаційної безпеки

Рівень вищої освіти - другий (магістерський)
Спеціальність - 125 «Кібербезпека та захист інформації»
Освітньо-професійна програма «Системи, технології та математичні методи кібербезпеки»

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Дмитро ЛАНДЕ
(підпис)
«__» _____ 2025 р.

ЗАВДАННЯ
на магістерську дисертацію здобувачу вищої освіти

_____ Дризі Артему Вадимовичу
(прізвище, ім'я, по батькові)

1. Тема роботи:

“Модель машинного навчання для виявлення зловмисної активності в журналах подій”

керівник роботи Барановський Олексій Миколайович, кандидат наук, доцент кафедри інформаційної безпеки

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

затверджені наказом по університету від «4» листопада 2025 р. № 4797-с

2. Термін подання здобувачем вищої освіти роботи «15» грудня 2025 р.

3. Вихідні дані до роботи

Науково-технічна література за темою дослідження, технічна документація, спеціалізоване програмне забезпечення.

4. Зміст роботи

Аналіз сучасних підходів до виявлення кіберзагроз, дослідження обмежень стандартних методів захисту та обґрунтування необхідності застосування проактивного пошуку загроз та машинного навчання для протидії продвинутим атакам.

Обґрунтування доцільності використання інструментарію Elastic Stack для детектування аномальної поведінки на основі машинного навчання. Проведення експериментальних досліджень ефективності виявлення зловмисної активності на прикладі скомпрометованого веб-сервера, а також оцінка точності роботи статистичних моделей.

5. Перелік ілюстративного матеріалу (із зазначенням плакатів, презентацій тощо)

Презентація до захисту магістерської дисертації

6. Дата видачі завдання 25 червня 2025 Календарний план

з/п	Назва етапів виконання магістерської дисертації	Термін виконання етапів магістерської дисертації	Примітка
	Визначення напрямку дослідження та затвердження теми	25.06.2025 - 09.07.2025	
	Збір та аналіз літературних джерел за темою роботи	10.07.2025 - 31.08.2025	
	Розробка теоретичних методів та моделей дослідження	01.09.2025 - 30.09.2025	
	Програмна реалізація та розробка компонентів системи	01.10.2025 - 31.10.2025	
	Проведення експериментальних досліджень та аналіз результатів	01.11.2025 - 15.11.2025	
	Оформлення пояснювальної записки до дисертації	16.11.2025 - 10.12.2025	
	Формування презентаційних матеріалів та графічної частини	11.12.2025 - 15.12.2025	
	Підготовка та попередній захист роботи на кафедрі	16.12.2025	
	Підготовка до захисту та захист магістерської дисертації	17.12.2025 - 24.12.2025	

Здобувач вищої освіти

(підпис)

Артем Дрига

(Власне ім'я, ПРІЗВИЩЕ)

Керівник роботи

(підпис)

Олексій Барановський

(Власне ім'я, ПРІЗВИЩЕ)

РЕФЕРАТ

Магістерська дисертація містить 86 сторінок, що складаються із 4-ох розділів, 7 ілюстрацій, та 30 посилань на джерела інформації.

Актуальність роботи: в умовах еволюції кіберзагроз та масового переходу зловмисників до використання продвинутих тактик, особливої уваги набуває питання неефективності класичних засобів захисту та реагування на інциденти по факту сповіщення про них. Традиційні інструменти, такі як IDS/IPS, часто не здатні виявити нові атаки, які не містять в собі відомих сигнатур. З огляду на це, потребується створення підходу реалізації моделей машинного навчання та тестування ефективності статистичних методів машинного навчання для виявлення аномалій у системних журналах. Це дозволить не лише розкрити сліпі зони традиційного моніторингу, а й допомогти в процесі проактивного пошуку загроз.

Зв'язок роботи з науковими програмами, планами, темами: магістерська дисертація виконується у межах науково-дослідних тем кафедри інформаційної безпеки НН ФТІ «КПІ ім. Ігоря Сікорського», зокрема в напрямі «Застосування машинного навчання у системах виявлення аномальної поведінки користувачів, поведінкового аналізу програмного коду, тощо». Також обрана тема відповідає державній науково-технічній програмі розвитку сфери кібербезпеки в Україні. Роль автора полягає у розробці підходу до створення моделей машинного навчання на основі Elastic Stack для використання у середніх та великих компаніях.

Мета дослідження: розробка та оцінка ефективності гібридного підходу, що поєднує детерміновані правила виявлення та методи машинного навчання, для автоматизованого виявлення зловмисної активності у журналах подій інформаційних систем.

Об'єкт дослідження: процес виявлення аномалій та аналізу подій інформаційної безпеки в журналах подій операційної системи Linux та веб-серверу.

Предмет дослідження: методи та моделі машинного навчання для виявлення аномальної поведінки окремих користувачів та процесів.

Наукова новизна одержаних результатів полягає у розробці гібридного підходу до детекції кіберзагроз, який поєднує детермінований аналіз на основі кореляційних правил з методами машинного навчання для виявлення аномалій у поведінці об'єктів.

Практичне значення одержаних результатів полягає в тому, що розроблений підхід може бути використаний для створення сценаріїв використання у операційних центрах безпеки (Security Operation Center), що дозволить автоматизувати роботу аналітиків та зменшити час реакції на інциденти.

Публікації: результати роботи були апробовані на XI International Scientific and Theoretical Conference «Sectoral research XXI: characteristics and features»

Ключові слова: машинне навчання, Elastic Stack, виявлення аномалій, веб-шелл, SIEM.

ABSTRACT

The master's thesis contains 86 pages, consisting of 4 chapters, 7 illustrations, and 30 references to sources of information.

Relevance of the study: in the context of the evolution of cyber threats and the mass transition of attackers to the use of advanced tactics, the issue of the inefficiency of classic protection means and incident response based on alerts becomes particularly pressing. Traditional tools, such as IDS/IPS, are often unable to detect new attacks that do not contain known signatures. Given this, there is a need to create an approach for implementing machine learning models and testing the effectiveness of statistical machine learning methods for detecting anomalies in system logs. This will not only uncover the blind spots of traditional monitoring but also assist in the process of proactive threat hunting.

Connection with scientific programs, plans, and topics: this master's thesis was conducted within the framework of research projects of the academic department of Information Security in IPT, «Igor Sikorsky KPI», specifically in the direction «Application of machine learning in systems for detecting anomalous user behavior, behavioral analysis of program code, etc.». The selected topic also aligns with the state scientific and technical program for the advancement of cybersecurity in Ukraine. The author's contribution consists in developing an approach to creating machine learning models based on the Elastic Stack for use in medium and large companies.

Purpose of the research: development and evaluation of the effectiveness of a hybrid approach that combines deterministic detection rules and machine learning methods for the automated detection of malicious activity in information system event logs.

Object of the research: the process of anomaly detection and information security event analysis in the event logs of the Linux operating system and a web server.

Subject of research: methods and machine learning models for detecting anomalous behavior of individual users and processes.

The scientific novelty of the obtained results lies in the development of a hybrid approach to cyber threat detection, which combines deterministic analysis based on correlation rules with machine learning methods to detect anomalies in object behavior.

The practical value of the obtained results lies in the fact that the developed approach can be used to create use-case scenarios in Security Operation Centers (SOC), which will automate the work of analysts and reduce incident response time.

Publications: the results of study were published at the XI International Scientific and Theoretical Conference «Sectoral research XXI: characteristics and features».

Keywords: machine learning, Elastic Stack, anomaly detection, web shell, SIEM.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ	10
ВСТУП.....	11
1 ОГЛЯД ЛІТЕРАТУРИ ТА ПОСТАНОВКА ЗАДАЧІ.....	13
1.1 Сучасні типи атак та проблеми з їх детектуванням	13
1.2 Сучасні підходи до детекції атак.....	22
Висновки до розділу 1	25
2 ПОРІВНЯЛЬНИЙ АНАЛІЗ КРИТЕРІЇВ ВИЯВЛЕННЯ АТАК.....	26
2.1 Формалізація задачі	26
2.2 Метрики виявлення атак	28
2.3. Функція вибору критеріїв	36
2.4 Порівняльний аналіз виявлення аномалій	38
Висновки до розділу 2.....	54
3 МОДЕЛЬ ВИЯВЛЕННЯ АНОМАЛІЙ	55
3.1 Концептуальна архітектура системи	55
3.2 Рівень збору та транспортування даних.....	56
3.3 Рівень нормалізації та збагачення даних.....	60
3.4 Рівень розподіленого зберігання та індексації даних	62
3.5 Рівень аналітики та детекції	64
3.6 Рівень візуалізації та реагування.	66
3.7 Запропонована архітектура та оцінка ефективності	68
Висновки до розділу 3.....	70
4 РЕАЛІЗАЦІЯ МОДЕЛІ	72

4.1 Запропонований стек технологій	72
4.2 Конфігурація середовища	75
4.3 Симуляції нормальної роботи	77
4.4 Конфігурація моделі виявлення аномалій.....	78
4.5 Результати роботи	79
Висновки до розділу 4.....	81
ВИСНОВКИ.....	82
ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ	84

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

APT - Advanced Persistent Threat (Розвинена стійка загроза);

AUC - Area Under Curve (Площа під кривою);

ELK - Elasticsearch, Logstash, Kibana (Стек інструментів);

FPR - False Positive Rate (Частка хибнопозитивних спрацювань);

LOLBins - Living off the Land Binaries (Легітимні файли ОС, що використовуються для атак);

LOTL - Living Off The Land (Використання стандартних інструментів системи для атаки);

LSTM - Long Short-Term Memory (Довга короткострокова пам'ять, тип нейромережі);

ML - Machine Learning (Машинне навчання);

PR-AUC - Precision-Recall Area Under Curve (Метрика точності-повноти);

SIEM - Security Information and Event Management (Система управління подіями безпеки);

SOC - Security Operation Center (Центр моніторингу безпеки);

SVM - Support Vector Machine (Метод опорних векторів);

ВСТУП

Актуальність теми: Сучасний етап розвитку інформаційних технологій характеризується експоненційним зростанням обсягів телеметрії, що генерується компонентами ІТ-інфраструктури. Журнали подій операційних систем, мережевого обладнання, хмарних середовищ та інструментів забезпечення кібербезпеки EDR містять критично важливу інформацію про стан інфраструктури, проте експоненційне зростання обсягу цих даних робить ручний аналіз неможливим.

В той же час, традиційні сигнатурні методи втрачають ефективність перед лицем новітніх атак, таких як застосування безфайлового шкідливого програмного забезпечення (Fileless Malware) або атак на ланцюги постачання (Supply Chains Attacks).

Враховуючи ці фактори, виникає об'єктивна потреба в переході від детермінованих правил до розробки моделей виявлення аномалій на основі статистичних алгоритмів, які зможуть ефективно працювати з великими об'ємом журналів подій. Використання методів машинного навчання (Machine Learning) дозволяє розробити таку модель, враховуючі будь-яку кількість вхідних даних.

Мета дослідження: розробка та оцінка ефективності гібридного підходу, що поєднує детерміновані правила виявлення та методи машинного навчання, для автоматизованого виявлення зловмисної активності у журналах подій інформаційних систем.

Об'єктом дослідження є журнали подій інформаційних систем на прикладі веб-серверу на основі Nginx.

Предметом дослідження є методи виділення інформативних ознак потенційних атак з журналів подій, методи машинного навчання та метрики оцінки якості виявлення кіберзагроз розробленою системою.

Для досягнення поставленої мети необхідно вирішити такі задачі:

1. Провести огляд сучасних векторів атак та проаналізувати відповідні артефакти, що залишаються після них в журналах подій;

2. Виконати аналіз існуючих методів виявлення аномалій та машинного навчання, визначити їх переваги та недоліки в контексті кібербезпеки;
3. Запропонувати удосконалений метод детекції та сформулювати вимоги до системи моніторингу;
4. Реалізувати програмний конвеєр обробки даних, що включає збір, нормалізацію та виділення ознак;
5. Провести експерименти із використанням розроблених моделей на основі реалізованого пайплайну;
6. Оцінити ефективність запропонованого підходу за допомогою метрик якості детекції.

Практична цінність роботи: результатом роботи є підхід на основі розробленого прототипу детектора аномалій та програмного конвеєр обробки даних. Запропонований підхід може бути використаний для створення сценаріїв використання у операційних центрах безпеки (Security Operation Center), що дозволить автоматизувати роботу аналітиків та зменшити час реакції на інциденти.

Структура роботи. Дисертація складається зі вступу, чотирьох розділів, висновків, списку використаних джерел.

1 ОГЛЯД ЛІТЕРАТУРИ ТА ПОСТАНОВКА ЗАДАЧІ

1.1 Сучасні типи атак та проблеми з їх детектуванням

Стрімка інтеграція інформаційних технологій у критичні бізнес-процеси призвела до парадоксальної ситуації: експоненційне зростання складності програмних комплексів прямо пропорційно збільшує поверхню атаки. В умовах сучасної цифрової інфраструктури, де веб-додатки виступають основним інтерфейсом взаємодії для користувачів, саме вони стають пріоритетною ціллю для зловмисників. Так, згідно зі звітом Verizon Data Breach Investigations Report 2024 [1], атаки на веб-додатки залишаються домінуючим вектором проникнення, складаючи значну частку від усіх підтверджених інцидентів витоку даних. Разом, аналіз інцидентів інформаційної безпеки останніх років демонструє тривожну тенденцію: методи наступальних операцій еволюціонують значно швидше, ніж засоби захисту.

Фундаментальна проблема полягає у вичерпанні потенціалу класичних, детермінованих підходів до виявлення загроз. Традиційні системи захисту, що спираються на сигнатурний метод, виявляються безсилими проти сучасних динамічних загроз. Аналітики ENISA (European Union Agency for Cybersecurity) у своєму прогнозі ландшафту загроз на 2025 рік підкреслюють, що зловмисники все частіше відмовляються від використання спеціалізованого шкідливого програмного забезпечення на користь атак на ланцюги постачання (Supply Chain Attacks) та експлуатації легітимних облікових записів, що ускладнює їх ідентифікацію стандартними засобами [2].

Сьогодні спостерігається зміщення вектору атак у бік використання легітимного функціоналу операційних систем та безфайлових методів виконання зловмисного коду. Зловмисник більше не приносить свій інструментарій, який можна заблокувати за вже відомими ознаками, натомість він використовує інструменти адміністратора проти самої системи. У такій ситуації межа між

нормальною роботою сервера та активною фазою кібератаки розмивається, а події безпеки губляться у шумі легітимного трафіку.

1.1.1 Концепція Living off the Land

Однією з найбільш складних для детектування технік, що набула значного поширення в сучасних кіберкампаніях рівня APT (Advanced Persistent Threat), є концепція **Living off the Land (LotL)**. Ця стратегія передбачає відмову зловмисників від завантаження в скомпрометовану систему специфічних шкідливих файлів (malware) на користь використання легітимного, попередньо встановленого програмного забезпечення та інструментів системного адміністрування [3].

Яскравим прикладом ефективності цієї методики є діяльність проурядової хакерської групи Volt Typhoon, атаки якої націлені на об'єкти критичної інфраструктури та залишалися непоміченими протягом тривалого часу саме завдяки маскуванню під штатну активність адміністраторів [4].

Інструменти, що використовуються в рамках цієї тактики, отримали назву **LOLBins** (Living off the Land Binaries) у середовищі Windows та **LOLScripts/GTFOBins** у середовищі Linux/Unix. До них відносяться:

- Інтерпретатори командного рядка (PowerShell, CMD, Bash, Zsh);
- Скриптові мови програмування, встановлені за замовчуванням (Python, Perl, PHP);
- Системні утиліти для роботи з мережею та завантаження файлів (curl, wget, netcat, certutil);
- Планувальники завдань (Task Scheduler, cron).

Згідно зі звітом компанії CrowdStrike, до переліку найбільш експлуатованих LOLBins у період 2021-2022 років увійшли [5]:

- Rundll32 (запуск DLL-бібліотек);
- Regsvr32 (реєстрація об'єктів COM);

- Msieхес (установники пакетів);
- Mshta (виконання HTA-файлів);
- Certutil (утиліта сертифікатів, часто використовується для декодування та завантаження пейлоадів);
- MSBuild та WMI (Windows Management Instrumentation).

Основна складність ідентифікації атак LotL полягає в подвійному призначенні використовуваних інструментів. З точки зору класичних засобів захисту, ці утиліти є довіреними компонентами операційної системи. Через це, виникають наступні проблеми при виявленні даного типу атак:

1. **Неефективність сигнатурного аналізу:** Файли LOLBins мають валідний цифровий підпис від вендора (наприклад, Microsoft) і коректну хеш-суму. Оскільки зловмисник не модифікує сам бінарний файл, а лише використовує його функціонал, антивірус не бачить загрози;
2. **Обхід білих списків (Whitelisting Bypass):** Політики безпеки, що базуються на дозволі запуску лише підписаних додатків (AppLocker, SELinux), не блокують LOLBins, оскільки вони є частиною ОС;
3. **Складність розрізнення контексту:** Технічно, запуск утиліти curl зловмисником для завантаження веб-шелу нічим не відрізняється від її запуску системним адміністратором для оновлення пакетів.

Внаслідок цього, детектування LotL зміщується з аналізу файлів на аналіз поведінки та контексту виконання. Сучасні підходи до виявлення таких загроз базуються на трьох ключових векторах:

- **Аналіз аргументів командного рядка:** Оскільки сам бінарний файл є легітимним, зловмисний намір приховано в параметрах запуску. Детектори шукають ознаки обфускації (Base64-кодування команд у PowerShell), завантаження даних із зовнішніх недовірених IP-адрес або використання специфічних прапорів (наприклад, -enc, -nop, -w hidden для PowerShell);

- **Аналіз дерева процесів (Process Lineage):** Відстеження родинних зв'язків процесів у вигляді «батько-дитина») Наприклад, запуск командного інтерпретатора `cmd.exe` або `powershell.exe` є підозрілим, якщо ініціатором процесу виступає офісний додаток (наприклад, `winword.exe` або `excel.exe`), що є типовою ознакою фішингової атаки з макросами, на відміну від запуску через `explorer.exe` (користувачем);
- **Аномалії частоти та часу:** Побудова профілю нормальної поведінки адміністратора. Якщо обліковий запис, який раніше використовував `net.exe` лише в робочий час для перевірки локальних налаштувань, раптово починає сканувати мережу вночі, це є індикатором компрометації.

1.1.2 Fileless Attacks

Еволюційним продовженням розвитку методів ухилення від виявлення стала поява класу загроз, відомих як **безфайлові атаки (Fileless Attacks)**. Їхньою визначальною характеристикою є здатність виконувати шкідливий код безпосередньо в оперативній пам'яті скомпрометованої системи, повністю оминаючи етап збереження виконуваного файлу на жорсткому диску. Цей підхід дозволяє зловмисникам обходити традиційні файлові антивіруси, які сканують статичні об'єкти на накопичувачах [6].

З точки зору архітектури операційних систем, такі атаки базуються на експлуатації легітимних механізмів управління пам'яттю та міжпроцесної взаємодії. Замість запуску окремого бінарного файлу, зловмисник впроваджує (ін'єктує) шкідливий код у віртуальний адресний простір вже запущеного, довіреного системного процесу (наприклад, `explorer.exe`, `svchost.exe` або `notepad.exe`). Ця група технік узагальнено відома як **Process Injection**. До найбільш небезпечних її різновидів відносять:

- **Process Hollowing:** створення легітимного процесу в призупиненому стані (suspended state), вивантаження його оригінального коду з пам'яті та заміна його на шкідливий код перед відновленням виконання [7];
- **Reflective DLL Injection:** завантаження динамічної бібліотеки (DLL) безпосередньо з пам'яті, міняючи стандартний завантажувач Windows (Windows Loader), що не залишає слідів реєстрації бібліотеки в системі [8].

Для підсистем захисту та адміністраторів безпеки безфайлові атаки створюють критичні виклики через низку факторів:

«Сліпота» традиційних засобів: Класичні антивірусні рішення та системи запобігання вторгненням (IPS) здебільшого базуються на сигнатурному аналізі файлових об'єктів. Відсутність файлу на диску унеможливорює порівняння його хеш-суми з базою відомих загроз;

Ефемерна природа доказів: Оскільки шкідливий код існує виключно в оперативній пам'яті, будь-яке перезавантаження системи призводить до повного знищення активної фази атаки. Це робить класичний криміналістичний аналіз (Forensics) практично неможливим без наявності попередньо налаштованих інструментів зняття дампу пам'яті (Live Memory Forensics).

Проте твердження про повну невидимість безфайлових атак є хибним у контексті сучасного моніторингу подій. Хоча атака не створює файлів, вона генерує специфічні подієві сліди (Event Artifacts) у журналах розширеного аудиту. Процес ін'єкції коду неминуче супроводжується викликами специфічних API-функцій ядра таких як:

- VirtualAllocEx,
- WriteProcessMemory,
- CreateRemoteThread тощо.

У логах це відображається як події доступу одного процесу до пам'яті іншого (Process Access) або створення віддаленого потоку.

Саме виявлення таких аномальних взаємозв'язків між процесами, коли, наприклад, офісний додаток намагається виділити пам'ять у системному процесі lsass.exe, може бути дієвим засобом детектування такого типу атак.

1.1.3 Supply Chain Attacks

Окремим вектором загроз, що набуває критичного значення в умовах глобального переходу до мікросервісної архітектури та поширення практики DevOps, є **атаки на ланцюг постачання (Supply Chain Attacks)**. На відміну від класичного сценарію прямого злому цільової інфраструктури (Direct Breach), дана стратегія передбачає компрометацію програмних продуктів, бібліотек залежностей або каналів оновлення третіх сторін ще до моменту їх потрапляння в інформаційну систему жертви [9]. Загалом, під ланцюгом постачання (Software Supply Chain) розуміють усю сукупність ресурсів, процесів та суб'єктів, задіяних у розробці та доставці програмного забезпечення: від репозиторіїв вихідного коду до механізмів доставки оновлень кінцевому користувачу.

Фундаментальна складність виявлення таких загроз полягає у експлуатації феномену «імпліцитної довіри» (Implicit Trust). Архітектура сучасних операційних систем та засобів захисту побудована на презумпції безпеки авторизованих джерел. Якщо виконуваний файл, інсталятор або скрипт оновлення має валідний цифровий підпис відомого вендора, система безпеки апріорі класифікує його активність як легітимну, часто виключаючи його з контуру поглибленого аналізу.

Зловмисники використовують цю архітектурну вразливість, модифікуючи вихідний код популярних бібліотек у публічних репозиторіях (наприклад, пакети npm, PyPI, RubyGems) або компрометуючи сервери збірки (Build Servers) розробників ПЗ. Реалізуються такі техніки як:

- **Dependency Confusion:** завантаження шкідливого пакету з публічного репозиторію замість внутрішнього приватного пакету з такою ж назвою [10];

- **Typosquatting:** створення шкідливих бібліотек зі схожими назвами на популярні пакети [11].

Коли адміністратор або автоматизована CI/CD система виконує стандартну процедуру оновлення, разом з корисним функціоналом завантажується та виконується прихований шкідливий код. Таким чином, компрометація ланцюга постачання перетворює довірені процеси на інсайдерські загрози.

З точки зору детектування на основі журналів подій, це створює унікальний виклик через те, що статичні правила безсилі, оскільки умовним зловмисником виступає легітимний процес оновлення (наприклад, `updater.exe` або `nrm`). Тому єдиним ефективним методом виявлення стає поведінковий аналіз аномалій, які повинні фіксувати не сам факт запуску програми, а відхилення у її поведінці:

- Чи почав довірений процес встановлювати з'єднання з новими, невідомими IP-адресами (C2-серверами)?
- Чи породив процес оновлення нетипові дочірні процеси (наприклад, запуск `cmd.exe` або `powershell.exe` після завершення завантаження)?
- Чи змінився патерн доступу до файлової системи?

Саме здатність виявляти такі девіації у поведінці формально довірених об'єктів є ключовим фактором виявлення такого типу атак.

1.1.3 Експлойти «нульового дня» (Zero-Day Exploits)

Найвищий рівень загрози у сучасній ієрархії кібератак **становлять експлойти для вразливостей «нульового дня» (Zero-Day Exploits)**. Це клас шкідливого програмного забезпечення або методик злому, що використовують програмні помилки чи архітектурні недоліки, про які на момент атаки ще невідомо вендору (розробнику) програмного забезпечення [12]. Відповідно, для них не існує офіційних виправлень (patches) або опублікованих бюлетенів безпеки CVE (Common Vulnerabilities and Exposures).

Фундаментальна проблема захисту від таких загроз полягає в асиметрії інформації: зловмисник володіє знанням про вразливість, тоді як сторона захисту діє в умовах повної невизначеності. Традиційні засоби захисту у вигляді мережевих системи запобігання вторгненням (NIDS/IPS) та антивірусів функціонують за реактивним принципом. Вони порівнюють вхідний потік даних або скановані файли з базою відомих сигнатур (патернів) атак.

У випадку атаки нульового дня множина відомих сигнатур для даної загрози є порожньою ($S_{signature} = \emptyset$). Це означає, що захисний механізм, аналізуючи мережевий пакет або файл із новим експлойтом, пропустить його як легітимний об'єкт, оскільки він формально не порушує жодного з існуючих правил фільтрації. Така вразливість може міститися будь-де:

- В ядрі операційної системи;
- В драйвері принтера;
- В сервісі SMB;
- В офісному додатку для перегляду документів тощо.

Однак, така невидимість для сигнатурного аналізу не означає відсутність слідів у телеметрії. Будь-яка успішна експлуатація вразливості (будь то переповнення буфера або ін'єкція команд) неминуче супроводжується відхиленнями від нормального профілю функціонування системи, які фіксуються в журналах подій. Характерними ознаками (артефактами) Zero-Day атак у логах загального призначення можуть бути:

- **Аномальна поведінка процесів:** Раптовий запуск інтерпретатора командного рядка (наприклад, cmd.exe або sh) від імені процесу, який зазвичай цього не робить (наприклад, процес обробки документів або калькулятор);
- **Ознаки нестабільності системи:** Поява записів про аварійне завершення служб (таких як Application Crash, Segmentation Fault) або помилки пам'яті безпосередньо перед встановленням підозрілого

з'єднання. Часто експлойт спрацьовує не з першого разу, викликаючи «падіння» сервісу.

- **Висока ентропія даних:** Використання шифрованого або обфускованого шел-коду (shellcode) призводить до підвищення ентропії в аргументах командного рядка або в тілі мережесих пакетів, що різко відрізняється від структури звичайних текстових даних;
- **Специфічні системні виклики:** Фіксація спроб доступу до захищених областей пам'яті (LSASS) або спроб ін'єкції коду в чужі процеси.

Особливості даного типу атак зумовлюють перехід з підходу пошуку конкретног збігу коду на реагування на девіацію (відхилення) спостережуваної поведінки системи від побудованої моделі норми.

Розглянуті сучасні типи кібератак - зокрема Living off the Land, безфайлові атаки, атаки на ланцюг постачання та експлойти нульового дня - демонструють спільну характерну рису: цілеспрямоване зменшення кількості явних ознак компрометації та максимальне маскування зловмисної активності під легітимні процеси. Усі ці класи атак орієнтовані на обходження сигнатурних механізмів захисту, маскування під легитимний шум та експлуатацію довірчих відносин, закладених в архітектуру сучасних інформаційних систем.

Аналіз показує, що традиційні підходи до захисту, які базуються на виявленні відомих шкідливих файлів, статичних сигнатур або заборонених шаблонів, є малоефективними проти зазначених загроз. Відсутність сторонніх виконуваних файлів, використання легітимних інструментів, валідних цифрових підписів та офіційних каналів оновлення призводять до того, що зловмисна активність залишається поза межами класичних механізмів контролю.

У таких умовах журнали подій стають одним з небагатьох універсальних джерел інформації, які відображають фактичну поведінку користувачів, процесів і сервісів у системі. Саме аналіз журналів подій дозволяє виявляти непрямі ознаки компрометації, відхилення від типової поведінки та приховані ланцюги дій, що не можуть бути ідентифіковані на рівні окремих подій. У зв'язку з цим доцільним є

детальний розгляд існуючих підходів до виявлення загроз у журналах подій, їх переваг та обмежень.

1.2 Сучасні підходи до детекції атак

Завдання детекції загроз у журналах подій полягає у своєчасному виявленні зловмисної активності на основі великого обсягу різнорідних подій. У практиці кіберзахисту застосовуються різні підходи, які можна умовно поділити на детерміновані, статистичні та інтелектуальні. Кожен з них має свої переваги та обмеження, що зумовлює доцільність їх комбінованого використання.

Правил-орієнтовані підходи базуються на використанні заздалегідь визначених шаблонів поведінки, які відповідають відомим технікам атак. Такі правила формалізують експертні знання фахівців з інформаційної безпеки та використовуються у SIEM-системах у вигляді кореляційних правил або стандартних форматів детекційних правил, зокрема SIGMA [13]:

```
sysmon_quarkspw_filedump.yml
1  title: QuarksPwDump Dump File
2  status: experimental
3  description: Detects a dump file written by QuarksPwDump password dumper
4  references:
5  |   - https://jpcertcc.github.io/ToolAnalysisResultSheet/details/QuarksPwDump.htm
6  author: Florian Roth
7  date: 2018/02/10
8  level: critical
9  logsource:
10 |   product: windows
11 |   service: sysmon
12 detection:
13 |   selection:
14 |     # Sysmon: File Creation (ID 11)
15 |     EventID: 11
16 |     TargetFileName: '*\AppData\Local\Temp\SAM-*.dmp*'
17 |     condition: selection
18 falsepositives:
19 |   - Unknown
20
```

Рисунок 1.1 - Приклад SIGMA правила

Кореляційні правила дозволяють поєднувати події з різних джерел і часових інтервалів, формуючи логічні ланцюги. Наприклад, комбінація подій автентифікації, зміни прав доступу та мережевої активності може свідчити про компрометацію облікового запису. Основними перевагами такого підходу є висока пояснюваність результатів та можливість швидкого впровадження. Водночас

ефективність правил значною мірою залежить від їх актуальності, а нові або модифіковані атаки часто залишаються непоміченими.

Статистичні підходи ґрунтуються на аналізі кількісних характеристик журналів подій. До них належать методи оцінки середніх значень, дисперсії, ковзних середніх, стандартних відхилень, а також методи контролю змін у часі. Детекція здійснюється шляхом порівняння поточних показників з історичними базовими значеннями або фіксованими порогами.

Такі методи є ефективними для виявлення різких відхилень, наприклад аномального зростання кількості запитів або спроб входу. Проте вони чутливі до сезонності, змін у режимах роботи системи та потребують регулярного коригування параметрів. Неправильно налаштовані пороги можуть призводити як до пропуску атак, так і до великої кількості хибних спрацювань [14].

Класичні алгоритми машинного навчання використовуються для автоматизованої класифікації журналів подій на нормальні та потенційно зловмисні. До таких методів належать

- логістична регресія;
- метод опорних векторів (SVM);
- Random Forest;
- XGBoost [15].

Перед навчанням моделей журнали подій перетворюються у набір ознак, які можуть відображати частоту подій, типи дій, часові інтервали та контекст взаємодії. Перевагою цих методів є здатність виявляти складні залежності між ознаками та адаптуватися до змін у поведінці системи. Однак вони потребують наявності репрезентативних розмічених даних, що в умовах реальної експлуатації часто є складним завданням. Крім того, зі зростанням кількості ознак ускладнюється інтерпретація результатів.

Водночас, підходи аномалійного виявлення орієнтовані на моделювання нормальної поведінки системи без явного опису атак. До них належать

- Isolation Forest;

- One-Class SVM;
- автоенкодери, які навчаються на історичних даних і визначають ступінь відхилення нових подій від сформованої моделі [16].

Ці методи особливо корисні для виявлення невідомих або рідкісних атак, зокрема внутрішніх загроз. Водночас вони мають обмежену пояснюваність і часто потребують додаткової фільтрації результатів, оскільки будь-яка нетипова, але легітимна активність може бути класифікована як загроза.

Оскільки журнали подій часто містять текстову або напівструктуровану інформацію, до їх аналізу застосовуються методи обробки природної мови. Підходи на основі TF-IDF та векторних подань дозволяють перетворювати текстові повідомлення у числові ознаки. Більш складні методи, зокрема LSTM та Transformer, застосовуються для аналізу послідовностей подій.

Послідовні моделі враховують порядок і контекст подій, що є критично важливим для детекції багатостадійних атак. Основними недоліками цих підходів є висока обчислювальна складність, вимоги до обсягів навчальних даних та складність практичного впровадження в реальному часі.

Гібридні системи поєднують правила та алгоритми машинного навчання в єдину архітектуру. Правила використовуються для виявлення відомих шаблонів атак, тоді як відповідні моделі на основі алгоритмів забезпечують адаптивність і здатність до виявлення нових загроз при відносній нескладності реалізації.

Такий підхід дозволяє зменшити кількість хибних спрацювань, підвищити точність детекції та зберегти пояснюваність результатів. Саме гібридні системи нині вважаються найбільш перспективним напрямом розвитку систем аналізу журналів подій, так як комбінація рішень дозволяє мінімізувати навантаження на аналітиків безпеки, автоматизуючи фільтрацію очевидних інцидентів та фокусуючи увагу на складних, нестандартних загрозах.

Висновки до розділу 1

У першому розділі виконано комплексний аналіз сучасного ландшафту кіберзагроз та методів їх виявлення в корпоративних інформаційних системах.

- Встановлено, що еволюція кіберзагроз змістилася в бік використання технік LotL, безфайлових атак та компрометації ланцюга постачання. Ключовою особливістю цих загроз є експлуатація довірених системних інструментів та легітимних каналів оновлення, що дозволяє зловмисникам обходити традиційні периметрові засоби захисту та антивірусне програмне забезпечення;
- Доведено, що класичні сигнатурні підходи та статичні правила кореляції вичерпали свій потенціал проти новітніх атак. Оскільки зловмисна активність все частіше мімікрує під штатні дії адміністраторів, а експлойти нульового дня не мають відомих сигнатур, детерміновані методи демонструють високий рівень пропуску атак;
- Визначено, що незважаючи на скритність сучасних атак, вони залишають специфічні поведінкові артефакти в журналах подій. Однак ефективний пошук цих слідів у ручному режимі є неможливим через гетерогенність форматів даних, надвеликі обсяги телеметрії та проблему «імпліцитної довіри» до системних процесів;
- Проведений аналіз обґрунтовує необхідність відмови від виключно реактивних сигнатурних методів на користь проактивного підходу. Напрямком подальшого дослідження обрано розробку гібридного методу для виявлення аномалій, що поєднує поведінковий аналіз та алгоритми машинного навчання для автоматизованого виявлення аномалій у поведінці користувачів та сутностей.

2 ПОРІВНЯЛЬНИЙ АНАЛІЗ КРИТЕРІЇВ ВИЯВЛЕННЯ АТАК

2.1 Формалізація задачі

З точки зору системного аналізу, процес моніторингу інформаційної безпеки можна представити як обробку потоку даних, що генерується інформаційною системою. Формально вхідні дані представляють собою множину подій $E = \{e_1, e_2, \dots, e_n\}$, де кожна подія e_i є вектором у багатовимірному просторі ознак $\mathcal{X}$. Вектор e_i складається з атрибутів

$$e_i = (t_i, s_i, a_i, o_i, r_i, c_i) \quad (2.1)$$

Де

- t_i - часовий штамп події;
- s_i - суб'єкт дії (користувач, процес або сервіс);
- a_i - тип виконаної дії;
- o_i - об'єкт впливу;
- r_i - результат події;
- c_i - контекстні характеристики (IP-адреси, ідентифікатори хостів, рівень привілеїв тощо).

Основною метою є побудова алгоритму (відображення) $A: \mathcal{X} \rightarrow \mathcal{Y}$, який зіставляє кожному вектору події e_i мітку класу $y_i \in \mathcal{Y}$. Залежно від наявності розмічених даних, задача може формулюватися у двох варіантах:

1. Бінарна класифікація (Supervised Learning): Якщо множина класів $\mathcal{Y} = \{0, 1\}$, де 0 - легітимна подія, а 1 - зловмисна активність. Цей підхід вимагає наявності репрезентативної вибірки атак на етапі навчання.
2. Виявлення аномалій (Anomaly Detection): В умовах відсутності повної інформації про всі можливі типи атак, задача зводиться до оцінки ступеня належності події до класу «норма». Алгоритм будує оцінку щільності

розподілу нормальних даних $P(x)$, і рішення приймається на основі порогового значення τ :

$$y_i = \begin{cases} \text{Аномалія, якщо } P(e_i) < \tau \\ \text{Норм, якщо } P(e_i) \geq \tau \end{cases} \quad (2.2)$$

Розробка та експлуатація такої системи пов'язана з рядом суттєвих обмежень, які необхідно враховувати при виборі моделі:

- **Дисбаланс класів (Class Imbalance):** У реальному трафіку частка зловмисних подій є вкрай малою ($< 0.1\%$) порівняно з легітимним фоном, що робить стандартні метрики точності неінформативними;
- **Зашумленість розмітки (Label Noise):** Історичні дані можуть містити помилки розмітки, коли пропущені атаки помилково маркуються як нормальні події, що знижує якість навчання моделі.
- **Дрейф концепції (Concept Drift):** Статистичні властивості як легітимного трафіку, наприклад зміна бізнес-процесів або оновлення ПЗ, так і патернів атак змінюються з часом, що призводить до деградації моделі та вимагає механізмів донавчання;
- **Вимоги до пояснюваності (Explainability):** Модель типу «чорна скринька» є неприйнятною для операційної діяльності SOC. Аналітик повинен розуміти, які саме ознаки вплинули на рішення системи про класифікацію події як загрози;

Виходячи з наведених обмежень, до розроблюваної системи висуваються такі вимоги:

- **Метрична якість:** Максимізація повноти виявлення для мінімізації пропуску атак, при утриманні рівня хибних спрацювань у межах, допустимих для обробки командою аналітиків;
- **Часова ефективність:** Система повинна забезпечувати обробку подій у режимі, наближеному до реального часу, щоб мінімізувати час перебування зловмисника в мережі;

- **Інтеграційна здатність:** Архітектура рішення повинна передбачати можливість інтеграції з існуючими платформами SIEM та системами автоматизованого реагування (SOAR) для передачі алертів та контекстних даних;

2.2 Метрики виявлення атак

При оцінюванні результатів роботи розробленої системи ключову роль відіграють метрики, які дозволяють кількісно оцінювати ефективність процесів виявлення загроз, а також при реалізації такої системи приймати обґрунтовані операційні та управлінські рішення.

При цьому, оцінка ефективності систем на основі алгоритмів машинного навчання у кібербезпеці вимагає відмови від стандартних підходів, що застосовуються у класичних задачах Data Science. При використанні базових метрик головною проблемою є екстремальний дисбаланс класів (Class Imbalance), коли кількість зловмисних подій є значно меншою за кількість легітимних. У реальному мережевому трафіку або журналах подій частка зловмисних подій часто становить менше 0.01%.

У таких умовах, метрика загальної точності (Accuracy) стає оманливою. Таке явище можна описати як «Парадокс точності» (Accuracy Paradox): тривіальна модель, яка класифікує абсолютно всі події як «безпечні», досягне точності 99.99%, але матиме нульову практичну цінність, оскільки пропустить усі атаки.

2.2.1 Категорії результатів виявлення

Основним інструментом для оцінки ефективності будь-якого класифікатора є **матриця помилок (Confusion Matrix)**. Це таблиця розмірністю $N \times N$, де рядки відповідають істинним класам (Actual), а стовпці - класам, що були передбачені моделлю (Predicted) [17]:

		Predicted	
		Positive	Negative
Actual	Positive	TP	FN
	Negative	FP	TN

Рисунок 2.1 - Матриця помилок

У контексті задачі виявлення вторгнень, де позитивний клас (Positive) - це наявність атаки, а негативний (Negative) - легітимна активність, елементи матриці інтерпретуються наступним чином:

- **True Positive (TP)** - зловмисні події, які були правильно виявлені системою як загрози;
- **False Positive (FP)** - помилки першого роду, тобто легітимні події, які помилково були класифіковані як зловмисні;
- **False Negative (FN)** - помилки другого роду, тобто зловмисні події, які система не виявила та класифікувала як легітимні;
- **True Negative (TN)** - легітимні події, які були правильно визначені як безпечні.

У контексті кібербезпеки ці категорії мають різну практичну вагу. Зокрема, False Negative асоціюються з пропущеними атаками і можуть призводити до суттєвих збитків, тоді як False Positive створюють надмірне навантаження на аналітиків та знижують ефективність реагування.

2.2.2 Метрики якості детекції

Метрика **точності детекції (Precision)** характеризує здатність моделі не генерувати хибні тривоги, тобто відображає частку коректних спрацювань серед усіх згенерованих сповіщень. Це дозволяє дати відповідь на питання "З усіх подій, які система назвала атаками, скільки насправді були атаками?".

Precision дорівнює відношенню кількості істинно позитивних спрацювань до суми істинно позитивних та хибнопозитивних спрацювань [18]:

$$Precision = \frac{TP}{TP + FP} \quad (2.3)$$

Метрика **повноти детекції (Recall)** визначає здатність системи виявляти реальні загрози. Це відповідь на питання: "Скільки реальних атак із тих, що сталися, система змогла виявити?".

Recall дорівнює відношенню кількості істинно позитивних спрацювань до суми істинно позитивних та хибнонегативних випадків [18]:

$$Recall = \frac{TP}{TP + FN} \quad (2.4)$$

Низький Recall означає, що значна частина атак залишається непоміченою. Recall повинна розглядатися як критично важлива метрика, оскільки пропущені атаки можуть мати серйозні наслідки для безпеки системи. Тому максимізація Recall є пріоритетом для критичних систем безпеки.

Метрика **F-оцінки (F1-Score)** - це гармонійне середнє між точністю та повнотою.

F1-Score дорівнює подвоєному добутку Precision та Recall, поділеному на їхню суму [18]:

$$F1 = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall} \quad (2.5)$$

Враховуючи проблему дисбалансу класів, виникає необхідність використання метрик, що фокусуються саме на здатності виявляти рідкісні, але критично важливі події.

Більшість алгоритмів виявлення аномалій на виході повертають не бінарну мітку класу (0 або 1), а безперервну величину - оцінку ймовірності належності до аномального класу $S \in [0,100]$ (Anomaly Score). Фінальне рішення про те, чи вважати подію інцидентом, залежить від обраного **порогу відсікання τ** .

Вибір цього порогу є критичним, оскільки він безпосередньо впливає на співвідношення між показниками точності та повноти детекції:

- **Зниження порогу** підвищує чутливість системи. У такому випадку зростає кількість виявлених атак (збільшується Recall), однак це супроводжується різким збільшенням кількості хибних спрацювань (зменшення Precision);
- **Підвищення порогу** робить систему більш вибірковою. Кількість хибних тривог зменшується, проте зростає ризик пропуску складних або маловиражених атак, що негативно впливає на повноту виявлення.

Для інтегральної оцінки якості моделі у всьому діапазоні можливих порогів використовуються графічні метрики, такі як **ROC-AUC (Area Under Receiver Operating Characteristic Curve)** та **PR-AUC (Area Under Precision-Recall Curve)**.

У випадку ROC-AUC крива помилок (ROC) відображає залежність між часткою вірних виявлень (TPR, True Positive Rate) [19]:

$$TPR(\tau) = \frac{TP(\tau)}{TP(\tau) + FN(\tau)} \quad (2.6)$$

та часткою хибних спрацювань (FPR, False Positive Rate):

$$FPR(\tau) = \frac{FP(\tau)}{TN(\tau) + FP(\tau)} \quad (2.7)$$

Якість моделі оцінюється як площа під цією кривою (AUC):

$$UC_{ROC} = \int_0^1 TPR(FPR) d(FPR) \quad (2.8)$$

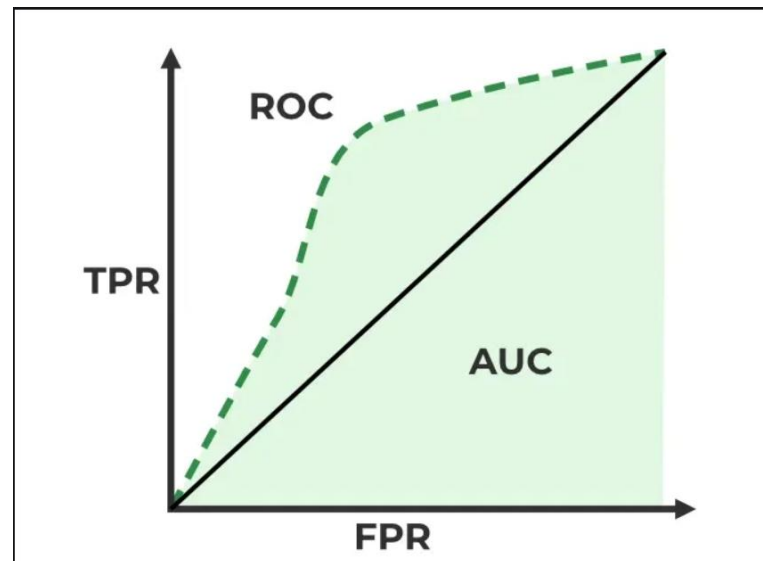


Рисунок 2.2 - Графічне представлення ROC-AUC

AUC_{ROC} дорівнює ймовірності того, що класифікатор оцінить випадково обраний позитивний екземпляр (атаку) вище, ніж випадково обраний негативний екземпляр (легітимну подію):

- $AUC = 0.5$: Випадкове вгадування (модель не працює);
- $AUC = 1.0$: Ідеальна класифікація.

Використання ROC-AUC на незбалансованих даних призводить до помилкових висновків. Причиною є знаменник у формулі частки хибних спрацювань який включає істинно хибні значення (TN). Це робить показник надто оптимістичним, оскільки велика кількість істинно негативних прикладів зменшує вплив хибних позитивних рішень на значення метрики.

Таким чином, високе значення ROC-AUC не завжди гарантує прийнятну операційну якість детекції у практичних умовах.

Precision-Recall (PR), на відміну від ROC-кривої, не враховує істинно негативні приклади, що робить її більш чутливою до якості виявлення рідкісних подій.

PR-крива визначається як графік залежності метрики **Precision** (вісь y) від метрики **Recall** (вісь x) при зміні порогу τ :

$$\text{Precision}(\tau) = \frac{\text{TP}(\tau)}{\text{TP}(\tau) + \text{FP}(\tau)} \quad (2.9)$$

$$Recall(\tau) = \frac{TP(\tau)}{TP(\tau) + FN(\tau)} \quad (2.10)$$

$$AUC_{PR} = \int_0^1 Precision(Recall) d(Recall) \quad (2.11)$$

На практиці, інтеграл обчислюється як зважена сума точності на кожному кроці зміни порогу (Average Precision):

$$AP = \sum_k (R_k - R_{k-1}) \cdot P_k \quad (2.12)$$

де:

- R_k (Recall) та P_k (Precision) - значення метрик при k -му порозі відсікання;
- $(R_k - R_{k-1})$ - приріст повноти (Recall) при переході до наступного порогу.

Зменшення площі під PR-кривою зазвичай свідчить про зростання кількості хибних спрацювань або пропущених атак, що безпосередньо впливає на операційну ефективність системи.

PR-AUC є узагальненим показником, який безпосередньо відображає компроміс між точністю та повнотою виявлення загроз. Саме тому PR-AUC вважається більш релевантною метрикою для оцінювання ефективності систем детекції в умовах дисбалансу класів.

2.2.3 Метрики узагальнювальної здатності

В умовах динамічного характеру кіберпростору, коли інфраструктура змінюється, оновлюються системи та додаються нові сервіси, розподіл подій в журналах подій з часом зміщується, і модель, яка працювала добре на історичних даних, може деградувати в експлуатації. На це також впливає виникнення нових типів атак, які були розглянуті в першому розділі.

Узагальнююча здатність (Generalization Capability) - ця характеристика визначає здатність моделі коректно класифікувати нові дані, які не були представлені під час навчання. У контексті кібербезпеки це означає здатність виявляти загрози, для яких ще не існує сигнатур.

Прикладом метрики оцінки узагальнюючої здатності є **Zero-Shot Detection Rate** - відсоток успішно виявлених атак класу, який був повністю відсутній у навчальному наборі даних, тобто визначається як відношення кількості успішно детектованих інцидентів невідомого класу TP_{zero} до загальної кількості таких атак N_{zero} :

$$ZSD = \frac{TP_{zero}}{N_{zero}} \cdot 100\% \quad (2.13)$$

2.2.4 Критерії ефективності

Методи виявлення загроз у журналах подій працюють у режимі близькому до реального часу, на великих обсягах даних та в умовах жорстких операційних вимог. Навіть модель з високими якісними та іншими метриками може бути непридатною, якщо:

- генерує надмірне обчислювальне навантаження;
- не вкладається у допустиму затримку (latency);
- потребує дорогої розмітки або частого перенавчання;
- не масштабується при зростанні потоку логів;
- складна у підтримці та зміні компонентів.

Тому при порівняльному аналізі методів доцільно враховувати не лише якість виявлення аномалій, а й сукупну вартість володіння (Total Cost of Ownership або TCO), продуктивність і масштабованість.

Однією з базових характеристик масштабованості системи аналізу журналів подій є **затримка обробки (latency)**, що відображає час від появи події у джерелі до моменту формування результату детекції та доступності алерту для аналітика. У кібербезпеці критично важливо оцінювати не лише середню затримку, а й “хвіст”

розподілу - зокрема перцентиль P95 (математична операція, яка відсікає 95% найменших значень у вибірці):

$$Latency_{P95} = quantile_{0.95}(Latency) \quad (2.14)$$

Це пояснюється тим, що пікові навантаження часто виникають саме під час інцидентів або масових змін в інфраструктурі, наприклад під час запуску оновлення, і якщо в ці моменти система видає великі затримки, детекція фактично запізнюється.

Окремим блоком інженерних критеріїв виступає **вартість експлуатації та підтримки (operational burden)**. У реальних середовищах найбільш дефіцитним ресурсом є не процесорний час, а час аналітиків та інженерів, які змушені розбирати сповіщення, виконувати їх сортування та підтримувати правила або моделі. Тому доцільно оцінювати не лише кількість сповіщень, а й сукупні витрати часу на їх обробку:

$$OpsBurden = (R_{alerts} \cdot T_{triage}) + T_{tuning} \quad (2.15)$$

де:

- R_{alerts} - середня кількість алертів за добу (*Alert Rate*);
- T_{triage} - середній час, що витрачає аналітик першої лінії на первинний аналіз одного сповіщення (*Mean Time to Triage*). Залежить від інтерпретованості моделі;
- T_{tuning} - час, що витрачають інженери на налаштування системи, оновлення правил та білих списків (*Maintenance Time*).

Якщо метод дає занадто багато спрацювань, це призводить до росту черги сповіщень, зниження уваги аналітиків та підвищення ризику пропуску реальної атаки

Однією з проблем використання вже розмічених даних для моделі є **вартість розмітки (Cost of Labeling)**. У кібербезпеці повноцінні мітки “атака/норма” або класи атак формуються шляхом аналізу інцидентів, співставлення подій з

результатами розслідування та експертної валідації. Це трудомісткий процес, а доступні мітки часто частково неповні або зашумлені (label noise).

Для математичної оцінки цих витрат, функцію сукупної вартості розмітки C_{label} можна представити наступним чином:

$$C_{label} = N_{events} \cdot C_{expert} \cdot (t_{base} + P_{mimicry} \cdot t_{context} + k_{noise} \cdot t_{verify}) \quad (2.16)$$

де:

- N_{events} - кількість подій, необхідних для формування репрезентативної навчальної вибірки;
- C_{expert} - вартість одиниці часу роботи кваліфікованого аналітика SOC;
- t_{base} - базовий час на аналіз очевидного інциденту;
- $P_{mimicry}$ - ймовірність того, що атака використовує техніки маскування (LotL, Living off the Land);
- $t_{context}$ - додатковий час, необхідний для глибокого контекстного аналізу складних атак;
- k_{noise} - коефіцієнт зашумленості даних, що вимагає повторної верифікації (t_{verify})

Особливо розмітка подій ускладнюється за рахунок новітніх методів атак, так як просунуті техніки зловмисників більшою мірою спрямовані на мімікрію під легітимні системні процеси та сервіси, що робить їх практично нерозрізнюваними від штатної активності адміністраторів без глибокого контекстного аналізу. Якщо метод потребує постійного поповнення розмітки, він стає важчим у підтримці та дорожчим в експлуатації, що в реальних умовах може переважити виграш у якості на історичних даних.

2.3. Функція вибору критеріїв

Для проведення формалізованого вибору оптимального методу детекції скористаємося методом адитивної згортки критеріїв. Цільова функція

ефективності $S(m_i)$ для методу m_i будується на основі п'яти ключових метрик, що були визначені раніше.

Нехай $K = \{k_1, k_2, k_3, k_4, k_5\}$ - множина критеріїв оцінювання:

- k_1 : PR-AUC (Area Under Precision-Recall Curve): в умовах дисбалансу класів ця метрика є найбільш репрезентативною;
- k_2 : ZSD (Zero-Shot Detection Rate);
- k_3 : Latency P95 (Затримка обробки);
- k_4 : OpsBurden (Операційне навантаження);
- k_5 : C_{label} (Вартість розмітки).

Оскільки метрики мають різну розмірність (відсотки, мілісекунди, грошові одиниці), виконуємо нормалізацію значень x_{ij} у діапазон $[0, 1]$:

Для критеріїв максимізації (PR-AUC, ZSD):

$$\widetilde{x}_{ij} = \frac{x_{ij} - \min_i(x_{ij})}{\max_i(x_{ij}) - \min_i(x_{ij})} \quad (2.17)$$

Для критеріїв мінімізації (Latency, OpsBurden, C_{label}):

$$\widetilde{x}_{ij} = \frac{\max_i(x_{ij}) - x_{ij}}{\max_i(x_{ij}) - \min_i(x_{ij})} \quad (2.18)$$

Інтегральна оцінка $S(m_i)$ розраховується як зважена сума нормованих критеріїв:

$$S(m_i) = w_1 \cdot \widetilde{PR-AUC} + w_2 \cdot \widetilde{ZSD} + w_3 \cdot \widetilde{Latency}_{P95} + w_4 \cdot \widetilde{OpsBurden} + w_5 \cdot \widetilde{C_{label}} \quad (2.19)$$

де $\sum w_j = 1$ - вагові коефіцієнти, що визначають пріоритетність вимог до системи.

Розподіл ваг $W = \{0.2, 0.3, 0.1, 0.2, 0.2\}$ базується на ієрархії вимог до системи виявлення просунутих кіберзагроз:

- $w_{ZSD} = 0.3$ (найвищий пріоритет): оскільки головним об'єктом дослідження є атаки типу сучасних типів атак, які за визначенням

відсутні у сигнатурних базах, здатність методу виявляти невідомі раніше загрози (ZSD) є критичною. Метод, що ідеально виявляє відомі віруси ($S \rightarrow 1$), але пропускає нові цільові атаки, не має практичної цінності для теми дисертації;

- $w_{PR-AUC} = 0.2$ (високий пріоритет): метрика точності залишається важливою, проте їй надано меншу вагу, ніж ZSD. В умовах виявлення новітніх загроз припустимо отримати декілька хибних спрацювань (False Positives), якщо це гарантує виявлення критичного інциденту;
- $w_{Ops} = 0.2$ та $w_{Cost} = 0.2$ (середній пріоритет): економічні та експлуатаційні показники мають значну вагу, оскільки складність налаштування (OpsBurden) та вартість розмітки даних (C_{label}) є головними бар'єрами для впровадження ML-систем виявлення аномалій у реальних SOC;
- $w_{Lat} = 0.1$ (Найнижчий пріоритет): аналіз журналів подій для виявлення складних патернів поведінки не вимагає мікросекундної реакції. Затримка обробки в межах декількох секунд або навіть хвилин є прийнятною для аналізу історичних логів.

2.4 Порівняльний аналіз виявлення аномалій

Вибір методу моделювання є компромісом між точністю детекції, обчислювальною складністю та вартістю впровадження. Враховуючи визначені метрики, проведемо критичний аналіз основних класів математичних моделей: від базової статистики до глибокого навчання.

2.4.1 Статистичні методи та порогові підходи

Статистичні методи є базовим класом підходів для виявлення аномалій у журналах подій. В основі підходу лежить припущення, що агреговані характеристики активності (наприклад, кількість подій певного типу за інтервал,

частота запитів, обсяг переданих даних, довжина сесій, ентропійні ознаки рядкових полів тощо) можуть бути описані статистичною моделлю “нормальної” поведінки (baseline). Аномалія, відповідно, визначається як значуще відхилення від очікуваних значень або розподілів.

Найпростішою реалізацією є детермінований пороговий аналіз. Нехай x_t - значення контрольної метрики у вікні t , а τ - заданий поріг. Тоді рішення детектора:

$$D(t) = \begin{cases} 1, \text{якщо } x_t > \tau \\ 0, \text{якщо } x_t \leq \tau \end{cases} \quad (2.20)$$

На практиці для згладжування короточасних сплесків використовується підхід ковзного вікна (Sliding Window), де аналізується не миттєве значення, а агрегована кількість подій за інтервал Δt :

$$\sum_{i=t-\Delta t}^t Event_i > T_{count} \quad (2.21)$$

Згідно з дослідженням [20], детермінований пороговий аналіз демонструє високу частоту помилок у динамічних середовищах через нездатність адаптуватися до змінних патернів телеметрії, тому розподіл оцінок виглядає наступним чином:

- PR-AUC (w_1): 0.2 (низька точність на нестационарних даних);
- ZSD (w_2): 0.1 (не виявляє "тихі" атаки на прикладі атаки Low-and-Slow);
- Latency (w_3): 1.0 (обчислення за $O(1)$);
- OpsBurden (w_4): 0.2 (потребує ручного перегляду порогів);
- Cost (w_5): 1.0 (не потребує розмітки даних).

Розрахунок інтегральної оцінки:

$$S_{thresh} = 0.2(0.2) + 0.3(0.1) + 0.1(1.0) + 0.2(0.2) + 0.2(1.0) = 0.41 \quad (2.19) \quad (2.22)$$

Для часткової адаптації порогу до дисперсії даних застосовується стандартизація через Z-score за наявності оцінок середнього μ та стандартного відхилення σ . Як критерій аномальності використовується правило «трьох сигм»: у

нормальному розподілі 99.73% легітимних подій лежать у діапазоні $[\mu - 3\sigma; \mu + 3\sigma]$. Таким чином, формула для Z-score набуває наступного вигляду:

$$Z = \frac{x - \mu}{\sigma}, \text{де Anomaly} \Leftrightarrow |Z| > 3 \quad (2.23)$$

Експерименти дослідників на реальних профілях навантаження показали, що Z-Score досягає максимального F1-score = 0.4, що є недостатнім для промислових систем [21]. Метод особливо неефективний для розподілів з "важкими хвостами" (Heavy-tailed).

Розподіл оцінок виглядає наступним чином:

- PR-AUC (w_1): 0.3 (низька);
- ZSD (w_2): 0.1 (реагує лише на амплітудні викиди);
- Latency (w_3): 0.9 (швидкий);
- OpsBurden (w_4): 0.3 (потребує ручного перегляду порогів);
- Cost (w_5): 1.0 (не потребує розмітки даних).

Розрахунок інтегральної оцінки:

$$S_{zscore} = 0.2(0.3) + 0.3(0.2) + 0.1(0.9) + 0.2(0.3) + 0.2(1.0) = 0.47 \quad (2.24)$$

Для виявлення змін у структурі подій відносно еталонного профілю використовують критерій узгодженості Пірсона (χ^2). Критерій узгодженості Пірсона дозволяє аналізувати структурну відповідність журналів подій еталонній моделі. Метод перевіряє нульову гіпотезу H_0 про те, що спостережуваний емпіричний розподіл подій у поточному вікні відповідає еталонному профілю. Критерій Пірсона розраховується як сума нормалізованих квадратів відхилень:

$$\chi^2 = \sum_{i=1}^k \frac{(O_i - E_i)^2}{E_i} \quad (2.25)$$

де:

- O_i (Observed) - спостережувана кількість подій категорії i у поточному вікні;
- E_i (Expected) - очікувана кількість подій цієї категорії згідно з історичним профілем ($E_i = N \cdot p_i$, де p_i - ймовірність категорії в нормі).

Робота "An anomaly detection technique based on a chi-square statistic " [22] підтверджує ефективність методу для стабільних систем, проте в умовах дрейфу концепції точність методу стрімко падає без постійного оновлення профілю E_i .

Розподіл оцінок виглядає наступним чином:

- PR-AUC: 0.4 (середня на стабільних даних);
- ZSD: 0.1 (не виявляє нові типи атак, що не порушують частотний розподіл);
- Latency: 0.8 (швидкий);
- OpsBurden: 0.1 (критично високий - ручне створення профілів).
- Cost (w_5): 1.0 (не потребує розмітки даних).

Розрахунок інтегральної оцінки:

$$S_{\chi^2} = 0.2(0.4) + 0.3(0.1) + 0.1(0.8) + 0.2(0.1) + 0.2(1.0) = 0.41 \quad (2.26)$$

2.4.2 Методи машинного навчання з учителем

Цей клас алгоритмів апроксимує невідому залежність $f: X \rightarrow Y$ на основі розміченої навчальної вибірки $D = \{(x_i, y_i)\}_{i=1}^N$, де $x_i \in R^d$ - вектор ознак, а $y_i \in \{0,1\}$ - мітка класу (норма/атака).

Для вирішення проблеми класифікації у просторах високої розмірності ефективним є метод опорних векторів (SVM): метод пошуку оптимальної розділяючої гіперплощини, що максимізує зазор (margin) між класами.

Задача зводиться до мінімізації норми вагового вектора w з умовою штрафу за помилки класифікації (ξ_i):

$$\min_{w,b,\xi} \left(\frac{1}{2} \|w\|^2 + C \sum_{i=1}^N \xi_i \right) \quad (2.27)$$

при обмеженні: $y_i(w^T \phi(x_i) + b) \geq 1 - \xi_i, \quad \xi_i \geq 0$

Для нелінійного розділення використовується "kernel trick" з радіальною базисною функцією (RBF):

$$K(x_i, x_j) = \exp \left(-\gamma \|x_i - x_j\|^2 \right) \quad (2.28)$$

Дослідження показують високу обчислювальну складність SVM ($O(N^3)$) на великих датасетах [23]. Хоча SVM ефективний на малих вибірках, він демонструє гіршу масштабованість порівняно з ансамблевими методами.

Розподіл оцінок виглядає наступним чином:

- PR-AUC: 0.85 (висока точність, але чутливий до шуму);
- ZSD: 0.1 (класифікує лише відомі патерни, обмежені опорними векторами);
- Latency: 0.6 (повільний на етапі передбачення при великій кількості опорних векторів);
- OpsBurden: 0.1 (складний підбір гіперпараметрів C, γ);
- Cost (w_5): 0.1 (вимагає повної розмітки).

Розрахунок інтегральної оцінки:

$$S_{SVM} = 0.2(0.85) + 0.3(0.1) + 0.1(0.6) + 0.2(0.3) + 0.2(0.1) = 0.34 \quad (2.29)$$

Водночас, щоб покращити стійкість до шуму, притаманного статистичним методам, застосовують Random Forest - ансамблевий метод, що базується на беггінгу (Bagging) множини вирішальних дерев $h(x)$.

Критерій розбиття вузла обирається шляхом мінімізації ентропії інформації або індексу Джині. Для індексу Джині $G(p) = \sum p_k(1 - p_k)$:

$$\Delta = G(\text{parent}) - (w_{\text{left}}G(\text{left}) + w_{\text{right}}G(\text{right})) \quad (2.30)$$

Фінальне рішення приймається шляхом мажоритарного голосування B дерев:

$$\hat{y} = \text{mode}\{h_1(x), h_2(x), \dots, h_B(x)\} \quad (2.31)$$

У дослідженні “Comparison of Selected Machine Learning Techniques for Cyberattack Anomaly Detection” [24] Random Forest показав найкращий результат серед усіх методів на датасеті IoT-23 з точністю **99.9%**. Проте, метод страждає від перенавчання на специфічних атаках і є безпорадним проти Zero-day загроз, які відсутні у навчальній вибірці (ZSD низький).

Розподіл оцінок виглядає наступним чином:

- PR-AUC: 0.99 (еталонна точність на відомих атаках);
- ZSD: 0.2 (не може екстраполювати правила на нові класи атак);

- Latency: 0.9 (дуже швидкий завдяки паралелізації дерев) ;
- OpsBurden: 0.4 (стійкий до викидів, менше налаштувань) ;
- Cost (w_5): 0.1 (вимагає повної розмітки).

Розрахунок інтегральної оцінки:

$$S_{RF} = 0.2(0.99) + 0.3(0.2) + 0.1(0.9) + 0.2(0.4) + 0.2(0.1) = 0.448 \quad (2.32)$$

Подальшим розвитком алгоритмів навчання з вчителем є метод градієнтного бустингу (XGBoost) - метод послідовної побудови ансамблю, де кожне наступне дерево виправляє помилки попередніх. Має адитивну модель навчання.

Нехай $\hat{y}_i$ - передбачення моделі для вхідного вектора x_i . Модель являє собою ансамбль з K адитивних функцій:

$$\hat{y}_i = \sum_{k=1}^K f_k(x_i), \quad f_k \in \mathcal{F} \quad (2.33)$$

де $\mathcal{F}$ - простір регресійних дерев (CART). На кожній ітерації t додається нове дерево f_t , яке мінімізує цільову функцію:

$$\widehat{y_i^{(t)}} = \widehat{y_i^{(t-1)}} + f_t(x_i) \quad (2.34)$$

Головною особливістю XGBoost є введення штрафного доданку Ω безпосередньо у функцію втрат:

$$\mathcal{L}^{(t)} = \sum_{i=1}^n l\left(y_i, \widehat{y_i^{(t-1)}} + f_t(x_i)\right) + \Omega(f_t) \quad (2.35)$$

де l - диференційовна опукла функція втрат (наприклад, LogLoss), а регуляризатор визначається як:

$$\Omega(f) = \gamma T + \frac{1}{2} \lambda \|w\|^2 \quad (2.36)$$

Тут T - кількість листків у дереві, w - вектор ваг у листках, γ (*гамма*) - штраф за створення нового листка (контроль складності дерева), λ - коефіцієнт L2-регуляризації (згладжування ваг).

Для оптимізації $\mathcal{L}^{(t)}$ використовується розкладання Тейлора другого порядку функції втрат в околі попереднього передбачення $\widehat{y_i^{(t-1)}}$:

$$\mathcal{L}^{(t)} \approx \sum_{i=1}^n \left[l(y_i, \widehat{y_i^{(t-1)}}) + g_i f_t(x_i) + \frac{1}{2} h_i f_t^2(x_i) \right] + \Omega(f_t) \quad (2.37)$$

де $g_i = \partial_{\widehat{y_i^{(t-1)}}} l(y_i, \widehat{y_i^{(t-1)}})$ - градієнт першого порядку, а $h_i = \partial_{\widehat{y_i^{(t-1)}}}^2 l(y_i, \widehat{y_i^{(t-1)}})$ - гессіан (градієнт другого порядку).

Об'єднавши індекси екземплярів i , що потрапляють у листок j , як множину I_j , можна переписати рівняння відносно ваг w_j :

$$\mathcal{L}^{(t)} \approx \sum_{j=1}^T \left[\left(\sum_{i \in I_j} g_i \right) w_j + \frac{1}{2} \left(\sum_{i \in I_j} h_i + \lambda \right) w_j^2 \right] + \gamma T \quad (2.38)$$

Оптимальне значення ваги w_j^* для фіксованої структури дерева знаходиться шляхом взяття похідної та прирівнювання до нуля:

$$w_j^* = - \frac{\sum_{i \in I_j} g_i}{\sum_{i \in I_j} h_i + \lambda} \quad (2.39)$$

Підставивши w_j^* назад у функцію втрат, отримуємо формулу структурної оцінки (Structure Score) дерева:

$$\mathcal{L}^*(q) = - \frac{1}{2} \sum_{j=1}^T \frac{\left(\sum_{i \in I_j} g_i \right)^2}{\sum_{i \in I_j} h_i + \lambda} + \gamma T \quad (2.40)$$

Для побудови дерева алгоритм жадібно шукає точку розбиття, яка максимізує приріст інформації (Gain). Формула приросту для розбиття листка на лівий (L) та правий (R) вузли виглядає так:

$$Gain = \frac{1}{2} \left[\frac{G_L^2}{H_L + \lambda} + \frac{G_R^2}{H_R + \lambda} - \frac{(G_L + G_R)^2}{H_L + H_R + \lambda} \right] - \gamma \quad (2.41)$$

Цей вираз математично показує баланс між покращенням функції втрат (перша частина формули) та штрафом за ускладнення моделі (γ). Якщо $Gain < 0$, розгалуження не виконується (pruning).

Завдяки використанню других похідних (h_i) у розкладанні Тейлора, алгоритм збігається значно швидше, ніж класичний GBM [25]. Результати порівняльних

тестів демонструють, що методи цього класу є безумовними лідерами за точністю класифікації відомих загроз.

Розподіл оцінок виглядає наступним чином:

- PR-AUC: 0.99 (максимальна точність на табличних даних);
- ZSD: 0.2 (Низька: алгоритм оптимізує ваги w_j виключно для відомих векторів x_i , і не має механізму екстраполяції на принципово нові розподіли ознак);
- Latency: 0.8 (висока швидкість передбачення, оскільки інференс зводиться до сумування значень з K дерев);
- OpsBurden: 0.2 (висока чутливість до гіперпараметрів регуляризації γ, λ та кроку навчання η , що вимагає складного налаштування);
- Cost (w_5): 0.1 (вимагає повної розмітки).

Розрахунок інтегральної оцінки:

$$S_{XGB} = 0.2(0.99) + 0.3(0.2) + 0.1(0.8) + 0.2(0.2) + 0.2(0.1) = 0.398 \quad (2.42)$$

2.4.3 Методи машинного навчання без учителя

Клас методів навчання без учителя спрямований на виявлення прихованих структур у даних X без використання розмічених міток Y . У контексті кібербезпеки задача зводиться до пошуку об'єктів x_i , які суттєво відрізняються від "нормального" розподілу генеральної сукупності.

Одним з основних алгоритмів навчання без вчителя є **Isolation Forest**. Алгоритм базується на гіпотезі, що аномалії є "рідкісними та відмінними" (Few and Different), тому в просторі ознак вони швидше відокремлюються (ізолюються) випадковими розбиттями, ніж нормальні екземпляри.

Нехай $X = \{x_1, \dots, x_n\}$ - вхідна вибірка з d -вимірному простору. Дерево ізоляції T будується рекурсивно шляхом поділу простору гіперплощинами, ортогональними до осей координат.

На кожному кроці j для вузла дерева:

- Випадковим чином обирається ознака $q \in \{1, \dots, d\}$;
- Випадковим чином обирається точка розбиття p з рівномірного розподілу: $p \sim U(\min(x_q), \max(x_q))$;

Множина X ділиться на дві підмножини:

$$X_L = \{x \in X \mid x_q < p\}, \quad X_R = \{x \in X \mid x_q \geq p\} \quad (2.36)$$

Процес триває до досягнення умови зупинки: $|X| = 1$ або досягнення граничної висоти дерева $h_{lim} = \lceil \log_2 n \rceil$.

Мірою аномальності є довжина шляху $h(x)$ від кореня до листка. Оскільки $h(x)$ залежить від розміру вибірки n , для порівнянності результатів виконується нормалізація відносно середньої довжини шляху в невіддаленому пошуку в бінарному дереві (BST).

Нормуючий коефіцієнт $c(n)$ визначається як:

$$c(n) = 2H(n-1) - \frac{2(n-1)}{n} \quad (2.43)$$

де $H(i)$ - гармонічне число, яке апроксимується через сталу Ейлера-Маскероні ($\gamma \approx 0.5772$):

$$H(i) \approx \ln(i) + \gamma \quad (2.44)$$

Фінальна оцінка $s(x, n)$ для екземпляра x розраховується як математичне сподівання довжини шляху по ансамблю з t дерев:

$$s(x, n) = 2^{-\frac{E(h(x))}{c(n)}} \quad (2.45)$$

Інтерпретація значень $s \in [0, 1]$:

- $\lim_{E(h(x)) \rightarrow 0} s = 1$ - аномалія (короткий шлях ізоляції);
- $\lim_{E(h(x)) \rightarrow n-1} s = 0$ - нормальний кластер;
- $s \approx 0.5$ - невизначений стан (шум).

Згідно з експериментальними даними дослідження Unsupervised Algorithms to Detect Zero-Day Attacks: Strategy and Application [26], алгоритм демонструє високу стійкість до перенавчання. Однак, в рамках іншого дослідження [27] на одному з

датасетів було виявлено зниження точності (Precision) в умовах щільного розташування кластерів атак (Masking Effect).

Розподіл оцінок виглядає наступним чином:

- **PR-AUC:** 0.70 (метод схильний до хибних спрацювань на складних, зашумлених даних, де нормальні об'єкти можуть мати характеристики викидів);
- **ZSD:** 0.8 (висока: метод не використовує знання про класи атак, тому ефективно ізолює загрози нульового дня виключно на основі їх геометричної віддаленості);
- **Latency:** 0.9 (дуже висока: алгоритм не обчислює відстані між точками, а інференс зводиться до проходження по дереву, що робить його одним із найшвидших ML-методів);
- **OpsBurden:** 0.7 (низьке навантаження: метод має мінімальну кількість гіперпараметрів - кількість дерев та розмір підвибірки);
- **Cost (C_{label}):** 1.0 (не потребує розмітки даних, оскільки належить до класу Unsupervised).

Розрахунок інтегральної оцінки:

$$S_{iForest} = 0.2(0.7) + 0.3(0.8) + 0.1(0.9) + 0.2(0.7) + 0.2(1.0) = 0.81 \quad (2.46)$$

Іншим прикладом навчання без учителя є **глибокі автоенкодер** (**Deep Autoencoders**) на базі штучних нейронних мереж, метою якого є апроксимація тотожного відображення $h_{W,b}(x) \approx x$ ерез прихований простір меншої розмірності (Bottleneck). Основна ідея полягає в тому, що мережа навчається стискати (кодувати) лише "нормальні" патерни даних, тому спроба відновити аномальний вхідний вектор призведе до значної помилки реконструкції.

Нехай $x \in R^d$ - вхідний вектор ознак. Автоенкодер складається з двох параметричних функцій:

Encoder (Кодер): Відображає вхідні дані у латентне представлення з меншої розмірності $h < d$:

$$z = f_{\theta}(x) = \sigma(Wx + b) \quad (2.47)$$

де $\theta = \{W, b\}$ - параметри кодера, σ - функція активації (наприклад, ReLU або tanh).

Decoder (Декодер): Відновлює вхідні дані з латентного коду:

$$\hat{x} = g_{\phi}(z) = \sigma'(W'z + b'') \quad (2.48)$$

де $\phi = \{W', b''\}$ - параметри декодера.

Задача навчання зводиться до мінімізації емпіричного ризику (функції втрат) на навчальній вибірці X_{train} , яка містить лише нормальний трафік:

$$\min_{\theta, \phi} \mathcal{L}(X) = \min_{\theta, \phi} \frac{1}{n} \sum_{i=1}^n \|x_i - g_{\phi}(f_{\theta}(x_i))\|_2^2 \quad (2.49)$$

Використання L_2 - норми (MSE) змушує мережу пріоритизувати найбільш значущі ознаки "норми".

Аномальність $A(x)$ визначається як евклідова відстань між оригінальним вектором x та його реконструкцією $\hat{x}$:

$$A(x) = \|x - \hat{x}\|_2 = \sqrt{\sum_{j=1}^d (x_j - \hat{x}_j)^2} \quad (2.50)$$

Рішення про детекцію приймається на основі порогу τ , який обирається як квантиль розподілу помилок на валідаційній вибірці (наприклад, P_{99}):

$$D(t) = \begin{cases} 1, & \text{якщо } A(x) > \tau \\ 0, & \text{якщо } A(x) \leq \tau \end{cases} \quad (2.51)$$

Згідно з дослідженням [28], автоенкодера демонструють вищу точність порівняно з класичними методами на складних, нелінійних даних, досягаючи Accuracy > 98%.

- **PR-AUC: 0.95** (дуже висока здатність моделювати складні нелінійні кореляції нормального профілю, що мінімізує помилкові спрацювання);

- **ZSD:** 0.9 (дуже висока: будь-який вектор атаки, що структурно відрізняється від норми, не зможе бути коректно стиснений кодером, що призведе до різкого зростання помилки реконструкції);
- **Latency:** 0.5 (середня: інференс вимагає виконання серії матричних множень $O(d \cdot h)$, що повільніше за проходження по дереву рішень, особливо на CPU);
- **OpsBurden:** 0.5 (висока складність проектування архітектури мережі: вибір кількості шарів, нейронів у "горлечку", функцій активації);
- **Cost (C_{label}):** 1.0 (навчається виключно на "чистому" трафіку без міток атак).

Розрахунок інтегральної оцінки:

$$S_{AE} = 0.2(0.95) + 0.3(0.9) + 0.1(0.5) + 0.2(0.5) + 0.2(1.0) = 0.81 \quad (2.52)$$

Сучасним методом виявлення аномалій, який є стандартом для сучасних систем UEBA (User and Entity Behavior Analytics), є **гібридний метод декомпозиції часових рядів та імовірнісного моделювання (Hybrid Time-Series Decomposition & Probabilistic Modeling) [29]**. Цей підхід представляє собою двоетапний математичний конвеєр (Pipeline), розроблений для усунення головного недоліку класичних статистичних методів - нездатності розрізняти аномалії та легітимні зміни навантаження (сезонність).

Метод розглядає задачу детекції як послідовність двох перетворень:

- Детерміноване перетворення (Decomposition): Очищення вхідного сигналу Y_t від прогнозованих компонент (тренду та сезонності) для отримання стохастичного залишку R_t ;
- Стохастичне оцінювання (Inference): Розрахунок ймовірності приналежності залишку R_t до нормального розподілу генеральної сукупності.

Така синергія дозволяє застосовувати суворий статистичний апарат (GMM) до нестационарних процесів реального часу.

Нехай Y_t - значення метрики (наприклад, кількість запитів за хвилину) у момент часу t . Використовується адитивна модель декомпозиції, що розкладає сигнал на три компоненти:

$$Y_t = T_t + S_t + R_t \quad (2.53)$$

- T_t (Trend) - довгостроковий напрямок зміни процесу. Оцінюється через метод локально зваженої регресії (Loess) або експоненційне згладжування:

$$T_t = \alpha Y_t + (1 - \alpha)(T_{t-1} + b_{t-1}) \quad (2.54)$$

- S_t (Seasonality) - циклічна компонента з періодом P (наприклад, $P = 24$ години), що відображає повторювані патерни:

$$S_t = \gamma(Y_t - T_t) + (1 - \gamma)S_{t-P} \quad (2.55)$$

- R_t (Residual) - залишковий компонент (шум), який містить стохастичні відхилення та потенційні аномалії:

$$R_t = Y_t - T_t - S_t \quad (2.56)$$

Суть методу полягає у тому, що аномалія шукається не у вихідному сигналі Y_t , а у очищеному залишку R_t . Це дозволяє ігнорувати легітимні піки навантаження, передбачені сезонністю S_t .

Для оцінки аномальності залишку R_t застосовується моделювання його функції густини ймовірності. Оскільки реальний трафік часто має мультимодальний розподіл, використовується суміш Гауссових розподілів (GMM):

$$P(R_t|\theta) = \sum_{k=1}^K \pi_k \mathcal{N}(R_t|\mu_k, \sigma_k^2) \quad (2.57)$$

Параметри моделі $\theta = \{\pi_k, \mu_k, \sigma_k\}$ оновлюються в режимі реального часу (Online Learning) за допомогою ЕМ-алгоритму, що дозволяє системі адаптуватися до дрейфу даних.

Для виявлення інсайдерських загроз порівнюється поведінка сутності u (користувача) з референсною групою G . Мірою аномальності є дивергенція Кульбака-Лейблера між розподілами ймовірностей:

$$D_{KL} (P_u || P_G) = \sum_x P_u(x) \cdot \ln \left(\frac{P_u(x)}{P_G(x)} \right) \quad (2.58)$$

Якщо D_{KL} перевищує поріг, сутність вважається аномальною відносно популяції, навіть якщо її абсолютні показники знаходяться в межах норми.

Через свої особливості, гібридний метод декомпозиції часових рядів та імовірнісного моделювання має декілька важливих переваг:

- Алгоритми STL-декомпозиції та оновлення параметрів GMM мають обчислювальну складність $O(1)$ або $O(K)$ на кожен новий екземпляр даних (де K - кількість компонент суміші, зазвичай < 5). Це дозволяє обробляти тисячі подій за секунду (EPS) в потоковому режимі. Це значно швидше за глибокі нейромережі, які вимагають важких матричних множень;
- Головна перевага методів навчання без учителя для аналізу часових рядів - автоматичне визначення періодичності (Periodicity Detection). Оператору не потрібно вручну вказувати, що "в суботу трафік менший" - модель вивчає це сама;
- Байєсівські моделі здатні виявляти інсайдерські загрози, для яких не існує сигнатур.

Це зумовлює високі оцінки:

- PR-AUC: 0.95 (дуже висока: фільтрація сезонності (S_t) майже повністю усуває хибні спрацювання, викликані робочими циклами);
- ZSD: 0.85 (висока: здатність виявляти "повільні" атаки типу Low-and-Slow, які змінюють кут нахилу тренду T_t , але не перевищують статичні пороги);
- Latency: 0.9 (висока: алгоритми STL та оновлення GMM мають складність $O(1)$ на кожен новий екземпляр даних, що підходить для потокової обробки);

- OpsBurden: 0.8 (низьке навантаження: модель автоматично визначає періодичність даних P та не вимагає ручного налаштування порогів);
- Cost (C_{label}): 1.0 (Працює на сирих журналах подій).

Розрахунок інтегральної оцінки:

$$S_{TS} = 0.2(0.95) + 0.3(0.85) + 0.1(0.9) + 0.2(0.8) + 0.2(1.0) = 0.895 \quad (2.59)$$

2.4.4 Вибір алгоритму

На основі розроблених метрик та розрахунку інтегральної функції ефективності $S(m)$, сформулюємо висновки щодо застосовності кожної групи методів для задач:

- Статистичні методи: ця група методів виявилася найменш ефективною. Головним обмежуючим фактором є припущення про стаціонарність процесів. У реальних мережах, де трафік має добову сезонність та тренди, статистичні методи генерують критичну кількість помилок першого роду, або вимагають ручного налаштування порогів, що неможливо масштабувати;
- Методи навчання з учителем: ці методи демонструють еталонну точність на відомих атаках, але отримали низьку інтегральну оцінку. Це зумовлено двома критичними недоліками у вигляді нездатністю виявляти нові вектори атак, відсутні у навчальній вибірці, та економічною недоцільністю створення розмічених датасетів для кожної унікальної інфраструктури;
- Методи навчання без учителя: ця група виявила найкращу здатність виявляти аномалії, Головні переваги - автономність (не потребують розмітки даних) та здатність виявляти аномалії будь-якої природи.

При цьому, гібридний метод декомпозиції часових рядів та імовірнісного моделювання вирішує проблему сезонності, забезпечуючи найкращий баланс між

точністю та автоматизацією. Саме через свої характеристики та отриману оцінку, його було обрано як фундамент системи виявлення зловмисної активності.

Таблиця 2.1 - Загальна таблиця оцінок для кожного методу

Категорія	Алгоритм	PR-AUC (w1)	ZSD (w2)	Lat (w3)	Ops (w4)	Cost (w5)	Інтегральна оцінка S(m)
Статистичні	Static Thresholds	0.20	0.10	1.0	0.2	1.0	0.410
	χ^2 Test	0.40	0.10	0.8	0.1	1.0	0.410
	Z-Score	0.30	0.20	0.9	0.3	1.0	0.470
З вчителем	SVM	0.85	0.10	0.6	0.3	0.1	0.340
	XGBoost	0.99	0.20	0.8	0.2	0.1	0.398
	Random Forest	0.99	0.20	0.9	0.4	0.1	0.448
Без учителя	Isolation Forest	0.70	0.80	0.9	0.7	1.0	0.810
	Autoencoders	0.95	0.90	0.5	0.5	1.0	0.810
Гібридні	Time-Series + Probabilistic	0.95	0.85	0.9	0.8	1.0	0.895

Висновки до розділу 2

У другому розділі було проведено системний аналіз об'єкта дослідження та існуючих методів виявлення аномалій. В результаті було зроблено наступне:

- Розроблено адитивну функцію ефективності, яка враховує баланс між точністю виявлення, повнотою охоплення загроз та обчислювальною складністю.
- За допомогою розробленої адитивної функції ефективності було проведено порівняльний аналіз декількох підходів до виявлення аномалій. Це дозволило кількісно обґрунтувати вибір гібридного методу декомпозиції часових рядів як найбільш ефективного, оскільки він забезпечує оптимальний баланс між повнотою детекції та мінімізацією обчислювальних витрат порівняно з існуючими рішеннями;
- Обґрунтовано необхідність побудови гібридної архітектури, яка поєднує кореляційні правила та методи машинного навчання для аналізу відхилень у залишкових компонентах часового ряду.

3 МОДЕЛЬ ВИЯВЛЕННЯ АНОМАЛІЙ

Базуючись на висновках щодо обмежень класичних методів навчання без учителя та неефективності статичних порогових правил, у даній роботі пропонується гібридний підхід до побудови системи детекції аномалій.

Запропонована концепція поєднує адаптивні можливості машинного навчання для виявлення невідомих загроз із детермінованою логікою кореляційних правил для фільтрації відомих патернів. Такий симбіоз дозволяє мінімізувати кількість хибних спрацювань при збереженні високої чутливості до нових векторів атак. При цьому, ключова ідея полягає в тому, щоб не обмежуватись аналізом окремих журналів подій, а перетворити потік подій у структуровані об'єкти спостереження (сесії, часові вікна або “сутність-орієнтовані” профілі), після чого оцінювати їх на предмет відхилення від нормальної поведінки.

3.1 Концептуальна архітектура системи

Для реалізації запропонованого гібридного методу виявлення загроз розроблено багаторівневу модульну архітектуру. Вона базується на принципах мікросервісної взаємодії, горизонтальної масштабованості та відмовостійкості. Система спроектована таким чином, щоб забезпечити повний конвеєр обробки даних (Data Pipeline): від моменту генерації події на кінцевому вузлі до прийняття рішення аналітиком SOC.

Архітектура складається з п'яти функціональних рівнів, кожен з яких виконує ізольований набір задач:

- Рівень збору та транспортування даних;
- Рівень нормалізації та збагачення;
- Рівень розподіленого зберігання;
- Рівень аналітики та детекції;
- Рівень візуалізації та реагування.

3.2 Рівень збору та транспортування даних

Архітектура підсистеми збору даних базується на використанні легковагових агентів збору (Lightweight Collectors), що розгортаються на кінцевих точках інфраструктури. Основною функцією таких агентів є безперервний моніторинг файлових журналів, первинна фільтрація подій та їх передача до центрального вузла агрегації. Критичною вимогою до реалізації є мінімізація споживання ресурсів хоста (Low Footprint), що забезпечує пріоритетність виконання бізнес-задач.

З точки зору системного аналізу, процес функціонування агента описується як система масового обслуговування (СМО). Нехай $\lambda(t)$ - миттєва інтенсивність генерації подій (events per second), а μ - максимальна пропускна здатність каналу відправки.

Для забезпечення стійкості до сплесків навантаження (Bursts) агент використовує внутрішню чергу (Internal Queue). Позначимо $Q(t)$ як поточну кількість подій у буфері, а K - його максимальну ємність. Динаміка стану черги описується диференціальним рівнянням:

$$\frac{dQ(t)}{dt} = \lambda(t) - \mu, \quad 0 \leq Q(t) \leq K \quad (3.1)$$

Система перебуває у стані стійкої рівноваги, якщо середня швидкість генерації подій на інтервалі T не перевищує пропускну здатність:

$$\frac{1}{T} \int_0^T \lambda(t) dt \leq \mu \quad (3.2)$$

Ризик переповнення буфера виникає у критичному стані, коли черга досягає максимальної ємності ($Q(t) = K$). У цьому випадку, якщо вхідний потік перевищує пропускну здатність каналу, надлишкові пакети відкидаються. Математично інтенсивність втрати даних $Loss(t)$ описується функцією максимуму:

$$Loss(t) = \max(0, \lambda(t) - \mu) \quad (49) \quad (3.3)$$

Ця формула доказує, що втрати дорівнюють різниці $\lambda(t) - \mu$, коли прихід перевищує відправку ($\lambda(t) > \mu$), і дорівнюють нулю в інших випадках.

Для мінімізації $Loss(t)$ реалізовано протокол зворотного тиску (Back-pressure): при досягненні порогу заповнення буфера (наприклад, $Q(t) \geq 0.9K$) агент адаптивно знижує частоту опитування файлів, надаючи пріоритет передачі вже накопичених даних.

Безпека транспортного рівня забезпечується відповідно до тріади CIA (Confidentiality, Integrity, and Availability) через захищений тунель:

- Confidentiality: Шифрування трафіку за протоколом TLS 1.3.
- Integrity & Authenticity: Взаємна автентифікація (mTLS) для запобігання підключенню неавторизованих агентів ("Rogue Agent").

Таким чином, рівень збору гарантує доставку телеметрії з математично обґрунтованою стійкістю до пікових навантажень.

Для суворого оцінювання надійності агента розглянемо процес обробки логів як систему масового обслуговування типу **M/M/1/K** (за нотацією Кендалла), де:

- **M**: Вхідний потік подій є пуассонівським (випадкові моменти появи журналів подій);
- **M**: Час обробки та відправки одного пакету розподілено експоненційно;
- **1**: Один канал відправки (одне TCP-з'єднання);
- **K**: Обмежена ємність черги (Memory Queue Limit).

Введемо коефіцієнт завантаження системи ρ (Traffic Intensity), який визначається відношенням швидкості генерації подій до пропускної здатності каналу:

$$\rho = \frac{\lambda}{\mu} \quad (3.4)$$

У стаціонарному режимі ймовірність P_n того, що в черзі знаходиться n пакетів, визначається геометричним розподілом:

$$P_n = \frac{1 - \rho}{1 - \rho^{K+1}} \rho^n \quad (3.5)$$

Критично важливою метрикою для системи безпеки є ймовірність відмови в обслуговуванні P_{loss} (тобто ймовірність втрати аудиту події через повний буфер). Для системи з обмеженою чергою K ця ймовірність дорівнює ймовірності того, що в системі вже знаходиться K заявок:

$$P_{loss} = P_K = \frac{1 - \rho}{1 - \rho^{K+1}} \rho^K \quad (3.6)$$

Аналіз стійкості:

- Якщо $\rho < 1$ (нормальний режим), $P_{loss} \rightarrow 0$ при збільшенні K ;
- Якщо $\rho > 1$ (DDoS або відмова мережі), черга заповнюється експоненційно швидко, і $P_{loss} \rightarrow 1 - \frac{1}{\rho}$.

Ця формула дозволяє аналітично розрахувати необхідний розмір буфера K для забезпечення заданого рівня надійності (наприклад, $P_{loss} < 10^{-6}$) при пікових навантаженнях.

Для запобігання ситуації переповненню черги ($\rho > 1$) агент реалізує механізм Back-pressure. Формалізуємо це як систему керування зі зворотним зв'язком.

Нехай $\lambda_{in}(t)$ - "сира" швидкість надходження логів від операційної системи, а $\lambda_{read}(t)$ - швидкість читання логів агентом (Harvester). Коли заповненість буфера $Q(t)$ перевищує поріг "м'якого ліміту" Q_{soft} , коефіцієнт передачі $f(Q)$ починає лінійно зменшуватися до нуля, поки черга не досягне критичної межі Q_{hard} :

$$f(Q) = \frac{Q_{hard} - Q(t)}{Q_{hard} - Q_{soft}} \quad (3.7)$$

Тоді фактична швидкість читання визначається як:

$$\lambda_{read}(t) = \lambda_{in}(t) \cdot \max(0, f(Q)) \quad (3.8)$$

Ця модель забезпечує плавне регулювання: якщо $Q(t) = Q_{soft}$, коефіцієнт $f = 1$ (читання на повну швидкість). Якщо $Q(t) \rightarrow Q_{hard}$, коефіцієнт $f \rightarrow 0$ (повна зупинка читання). Це дозволяє перекласти навантаження з оперативної пам'яті агента на дисковий буфер операційної системи, запобігаючи втраті даних.

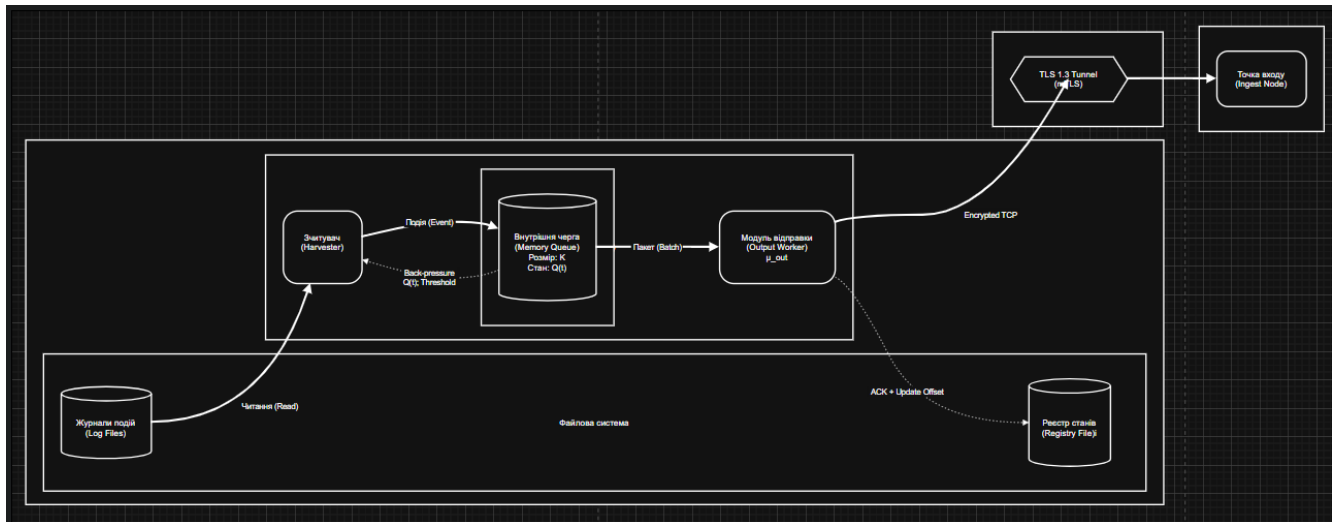


Рисунок 3.1 - Структурно-функціональна схема рівня збору даних

Опис схеми:

- Harvester: компонент, що моніторить файлові дескриптори. Він зчитує нові рядки та перетворює їх на структуровані об'єкти подій;
- Internal Queue (буфер): проміжна ланка, що згладжує нерівномірність генерації логів (Bursts). Саме тут розраховується P_{loss} ;
- Feedback Loop (червона лінія): візуалізація зворотного зв'язку. Якщо черга заповнена, Harvester призупиняє читання, змушуючи логи накопичуватися на рівні файлової системи (OS Disk Buffer), що є безпечнішим, ніж втрата в RAM.
- Transport: захищений канал передачі.

Окрім мережевої передачі, критичним є збереження стану читання файлів (File State Integrity). Нехай S_i - стан читання i -го файлу в момент часу t :

$$S_i(t) = \{\text{inode}_i, \text{offset}_i, \text{timestamp}_i\} \quad (3.9)$$

Агент періодично (з інтервалом Δt_{flush}) виконує операцію контрольної точки (Checkpointing), записуючи вектор станів S на диск у файл реєстру. У разі аварійного завершення роботи агента (Crash), відновлення позиції читання відбувається за правилом:

$$\text{offset}_{start} = \max(\text{offset}_{registry}, 0) \quad (3.10)$$

Це гарантує властивість **At-Least-Once Delivery** (доставка принаймні один раз), що є необхідною умовою для систем аудиту безпеки, допускаючи дублювання, які усуваються на наступних рівнях архітектури (дедуплікація за event.id).

3.3 Рівень нормалізації та збагачення даних

Рівень нормалізації та збагачення даних виконує перетворення неструктурованих сирих подій, отриманих від агентів, у канонічний формат (Common Schema), що є критичним для коректної роботи алгоритмів машинного навчання. Математично цей етап можна представити як послідовність операцій над множиною вхідних векторів.

Нехай E_{raw} - сира подія у вигляді текстового рядка. Процес парсингу - це функція відображення f_{parse} , яка перетворює рядок у структурований вектор ознак $x \in R^d$:

$$x = f_{parse}(E_{raw}) \quad (3.11)$$

Для вилучення значень використовується апарат регулярних виразів, що базується на теорії скінченних автоматів. Складність обробки одного повідомлення T_{proc} лінійно залежить від довжини рядка n та кількості правил вилучення m :

$$T_{proc} \approx O(n \cdot m) \quad (3.12)$$

Збагачення даних (Enrichment) - це операція додавання контекстуальних ознак до вектора x . Нехай D - зовнішній довідник (наприклад, списки репутації). Тоді збагачений вектор x' визначається як:

$$x' = x \cup \{v \mid v = \text{lookup}(x_i, D)\} \quad (3.13)$$

де x_i - ключ пошуку (наприклад, IP-адреса).

Для оптимізації продуктивності на цьому етапі застосовується фільтр Блума (Bloom Filter), який дозволяє миттєво визначити відсутність ключа в базі шкідливих об'єктів без звернення до диска. Ймовірність хибнопозитивного спрацювання f фільтра Блума при k хеш-функціях та m бітах пам'яті для n елементів становить:

$$f \approx (1 - e^{-kn/m})^k \quad (3.14)$$

Оскільки рівень нормалізації є обчислювально важким (CPU-intensive), його можна змодельовати як багатоканальну СМО (М/М/с).

Якщо λ - швидкість надходження подій, а μ_p - швидкість обробки подій одним потоком (worker), то загальна пропускна здатність рівня становить $c \cdot \mu_p$, де c - кількість виділених ядер процесора.

Умова відсутності черг на рівні нормалізації:

$$\rho_{norm} = \frac{\lambda}{c \cdot \mu_p} < 1 \quad (3.14)$$

Для візуалізації процесу розглянемо функцію трансформації як перетворення ентропії даних. Сирі логи мають високу ентропію $H(E_{raw})$ через неструктурованість. Після нормалізації ентропія структури зменшується, а інформаційна щільність зростає.

Таблиця 3.1 - Етапи трансформації та нормалізації даних у конвеєрі обробки подій

Етап	Стан даних (Вхід)	Операція	Стан даних (Вихід)
Парсинг (Parsing)	String (Сирий текст)	Regex	JSON Object (Ключ-Значення)

Кінець таблиці 3.1

Типізація (Typing)	Mixed Types	Кастинг типів	Typed Vector (float, int, keyword)
Збагачення (Enrich)	Local context	API / DB Lookup	Enriched Vector (+Geo, +Reputation)

На виході ми отримуємо нормалізований потік $X_{norm} = \{x'_1, x'_2, \dots, x'_n\}$, де кожен вектор має фіксовану розмірність d' .

3.4 Рівень розподіленого зберігання та індексації даних

Після нормалізації дані передаються на рівень зберігання, який базується на архітектурі розподіленого інвертованого індексу. Основним завданням цього рівня є забезпечення константної швидкості пошуку $O(1)$ або $O(\log n)$ при обсягах даних порядку терабайтів.

Розглянемо множину нормалізованих документів $D = \{d_1, d_2, \dots, d_n\}$. Кожен документ складається з термів (ознак) T . Інвертований індекс - це відображення $f_{index}: T \rightarrow \mathcal{P}(D)$, де кожному терму t ставиться у відповідність список ідентифікаторів документів (posting list) L_t :

$$L_t = \{id_i \mid t \in d_i\} \quad (3.15)$$

Ефективність пошуку за декількома ознаками (наприклад, IP - адреса та порт) зводиться до операції перетину множин:

$$R = L_{t1} \cap L_{t2} \cap \dots \cap L_{tk} \quad (3.16)$$

Використання алгоритмів стиснення (наприклад, Roaring Bitmaps) дозволяє виконувати ці операції на рівні процесорних команд, що мінімізує затримку (*Latency*).

Для горизонтального масштабування індекс I розділяється на N шардів. Розподіл документів по шардах здійснюється за допомогою хеш-функції H :

$$shard_id = H(routing_key) \pmod{N} \quad (3.17)$$

Це забезпечує рівномірне завантаження вузлів кластера (Load Balancing). Інтенсивність запитів до одного вузла λ_{node} при загальній інтенсивності λ становить:

$$\lambda_{node} \approx \frac{\lambda}{N} \quad (3.18)$$

Оскільки обсяг пам'яті M обмежений, а потік даних $\lambda(t)$ безперервний, використовується стратегія ротації індексів. Загальний обсяг даних V за час T становить:

$$V = \int_0^T \lambda(t) \cdot size(x) dt \quad (3.19)$$

де $size(x)$ - середній розмір одного структурованого запису.

Для оптимізації вартості зберігання ($Cost$) застосовується багаторівнева модель:

- Hot Tier: $V_{hot} \propto \frac{1}{SSD\ Speed}$, де дані доступні для миттєвого аналізу;
- Cold Tier: $V_{cold} \propto Compression\ Ratio$, де дані зберігаються у стисненому вигляді.

Швидкість деградації продуктивності пошуку P залежить від кількості сегментів в індексі. Процес злиття сегментів (Merging) описується логарифмічною складністю, що дозволяє підтримувати стабільну швидкість доступу при експоненціальному зростанні обсягу бази даних. Це досягається шляхом фонові консолідації малих сегментів у більші структури, що мінімізує кількість звернень до дискової підсистеми під час виконання складних аналітичних запитів.

Для забезпечення доступності ($Availability$) кожен шард має R реплік. Ймовірність втрати даних P_{fail} при ймовірності виходу з ладу одного вузла p становить:

$$P_{fail} = p^{R+1} \quad (3.20)$$

При $R = 2$ та $p = 0.01$, ймовірність втрати даних стає критично малою (10^{-6}), що відповідає вимогам до систем корпоративного рівня.

Математичні моделі інвертованого індексу, шардування та реплікації, базуються на наборі системних параметрів. Саме правильний вибір цих констант дозволяє досягти балансу між швидкістю обробки вхідного потоку даних λ та надійністю їх зберігання P_{fail} .

Головні системні параметри наведено у таблиці 3.2.

Таблиця 3.2 - Параметри конфігурації рівня зберігання та їх вплив на характеристики системи

Параметр	Позначення	Вплив на систему
Кількість шардів	N	Визначає здатність до горизонтального масштабування та паралелізму запису
Фактор реплікації	R	Визначає рівень відпорності до відмов та ймовірність збереження даних
Інтервал оновлення	Δt_{idx}	Визначає баланс між навантаженням на CPU та швидкістю доступу до нових даних

3.5 Рівень аналітики та детекції

Цей рівень є інтелектуальним ядром системи. Відповідно до висновків Розділу 2, для забезпечення максимальної ефективності виявлення загроз реалізовано **гібридний підхід**. Він поєднує детермінований аналіз (кореляційні правила) для відомих атак та імовірнісне моделювання для виявлення невідомих аномалій

Детермінований аналіз базується на пошуку специфічних патернів у ковзному часовому вікні W . Для спрощення моделі кореляційне правило представимо як індикаторну функцію I_r , яка діє на підмножину подій $s \subseteq W$:

$$A_{rule}(W) = \max_{s \subseteq W} (I_r(s) \cdot \text{Severity}_r) \quad (3.21)$$

де $I_r(s) \in \{0,1\}$ набуває значення 1 лише тоді, коли сукупність подій s задовольняє логічному предикату $\Phi(s)$, що описує сигнатуру атаки. Параметр Severity_r позначає експертну вагу (критичність) даного правила. Такий підхід дозволяє лінеаризувати логіку детекції та легко інтегрувати її в загальну оцінку ризику.

Обраний метод декомпозиції часових рядів передбачає, що вхідний сигнал $x(t)$ є адитивною сумою трьох компонент:

$$x(t) = T(t) + S(t) + e(t) \quad (3.22)$$

Для знаходження випадкового залишку $e(t)$, який містить аномальну складову, ми застосовуємо оператор декомпозиції:

$$e(t) = x(t) - [T(t) + S(t)] \quad (3.23)$$

Для оцінки ступеня аномальності залишку $e(t)$ використовується Z-модифікований показник або відхилення за квантильною функцією.

Нехай $F(e)$ - функція розподілу ймовірностей значень залишку в нормальному стані (навчальна вибірка). Тоді імовірнісна оцінка аномалії P_{score} визначається як:

$$P_{score}(t) = |2 \cdot F(e(t)) - 1| \quad (3.24)$$

Ця формула масштабує результат у діапазоні $[0, 1]$, де:

- $P_{score} \rightarrow 0$ означає, що значення знаходиться в медіані розподілу (норма);
- $P_{score} \rightarrow 1$ означає, що значення знаходиться на "хвостах" розподілу (екстремальна аномалія).

Підсумковий індикатор тривоги $Alert(t)$ розраховується як зважена сума результатів обох методів. Для усунення розбіжностей у масштабах використовується нормалізація:

$$Alert(t) = w_1 \cdot A_{rule}(W) + w_2 \cdot \int_{t-\Delta t}^t P_{score}(\tau) d\tau \quad (3.25)$$

Другий доданок інтегрує імовірнісну оцінку на невеликому проміжку часу Δt , що дозволяє системі ігнорувати поодинокі викиди ("шум") і реагувати на сталі аномальні тренди.

Синтез результатів детермінованого аналізу за правилами та імовірнісної оцінки аномальності залишку часового ряду формалізується у вигляді дворівневого конвеєра прийняття рішень. Структурна схема цього процесу, що відображає шлях трансформації подій у пріоритезовані сповіщення, представлена на рисунку 3.2:

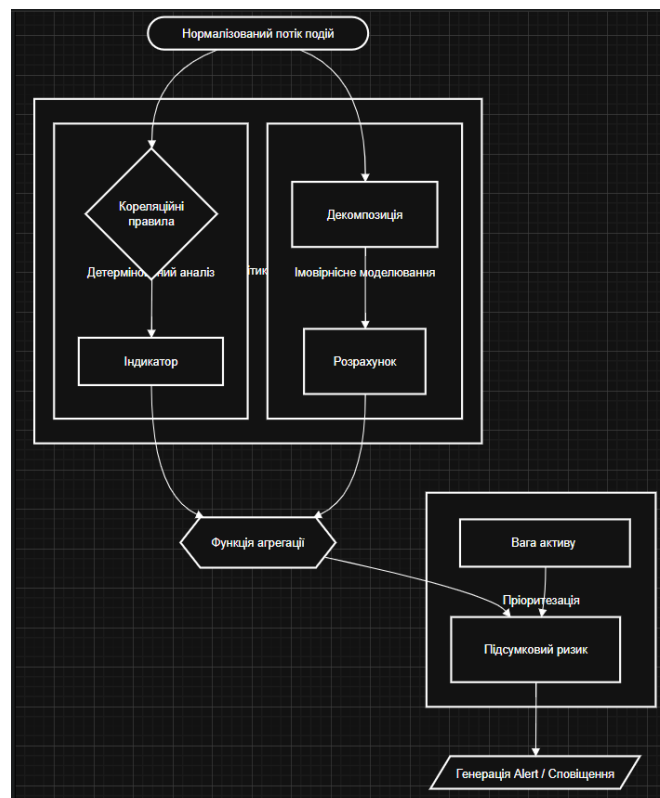


Рисунок 3.2 - Логічна схема гібридної детекції

3.6 Рівень візуалізації та реагування.

Рівень розподіленого зберігання та індексації даних є термінальною ланкою архітектури, яка трансформує результати аналітичної обробки у форму, придатну для прийняття рішень аналітиком кібербезпеки або автоматизованими системами стримування (SOAR). Математична модель цього рівня базується на теорії

ранжування та теорії масового обслуговування для оптимізації когнітивного навантаження на аналітика.

Оскільки потік аномалій $A(t)$ може бути надлишковим, вводиться функція ранжування інцидентів за рівнем ризику R_{score} . Вона враховує не лише імовірність аномалії, але й критичність вузла (активу). Нехай W_{asset} - вага активу, визначена через цінність даних, а V_{vuln} - коефіцієнт вразливості системи. Тоді підсумковий показник ризику інциденту I_r визначається як:

$$I_r = Alert(t) \cdot \frac{W_{asset}}{1 + e^{-\beta(V_{vuln} - \theta)}} \quad (3.26)$$

де логістична складова враховує "експоненціальне зростання" небезпеки при перевищенні порогу вразливості θ з коефіцієнтом агресивності β . Такий підхід дозволяє автоматично виносити на передній план тільки ті події, що мають критичний вплив на бізнес-процеси.

Для усунення проблеми "лавини сповіщень" при однотипних атаках застосовується алгоритм часової агрегації. Якщо система фіксує множину подій $E = \{e_1, e_2, \dots, e_n\}$ з однаковим ідентифікатором загрози у вікні Δt , кількість генерованих сповіщень N_{alert} обчислюється за логарифмічною шкалою:

$$N_{alert} = 1 + \lfloor \log_y(n) \rfloor \quad (3.27)$$

Це означає, що перше сповіщення генерується миттєво, а наступні - лише при експоненціальному зростанні кількості подій (наприклад, кожне 10-те, 100-те тощо). Це забезпечує інформативність без перевантаження каналів зв'язку.

Ефективність реагування оцінюється через мінімізацію втрат L . Час реакції системи T_{res} складається з часу детекції T_{det} та часу виконання сценарію блокування T_{act} :

$$L(T_{res}) = \int_{t_0}^{t_0 + T_{det} + T_{act}} \lambda_{loss}(u) du \quad (3.28)$$

Де $\lambda_{loss}(u)$ - швидкість витоку або пошкодження даних. Для автоматизації реагування впроваджується функція вибору контрзаходу C . Математично це задача багатокритеріальної оптимізації:

$$\text{find } C: \min(L(C) + \text{Cost}(C)) \quad (3.29)$$

Система обирає таку дію (наприклад, ізоляція хоста або блокування IP), яка мінімізує збитки при мінімальних витратах на переривання бізнес-процесів.

3.7 Запропонована архітектура та оцінка ефективності

Запропонована архітектура базується на принципі багатоешелованого захисту, де кожен рівень виконує специфічну математичну трансформацію даних, поступово знижуючи ентропію (невизначеність) та підвищуючи інформаційну цінність подій.

Процес функціонування системи можна представити як композицію функцій F , де вихід попереднього рівня є входом для наступного:

$$\text{Output} = f_{vis} \circ f_{ana} \circ f_{idx} \circ f_{norm} \circ f_{coll}(\text{Raw_Logs}) \quad (3.30)$$

- Рівень збору (f_{coll}): Фільтрує потік через модель M/M/1/K, гарантуючи цілісність $P(Q < K)$.
- Рівень нормалізації (f_{norm}): Проектує текстові дані у векторний простір R^d ;
- Рівень аналітики (f_{ana}): Реалізує гібридну функцію $\text{Alert}(t)$, що поєднує детерміновані правила та імовірнісну декомпозицію $T + S + e$;
- Рівень реагування (f_{vis}): Ранжує результат через функцію ризику I_r .

Для оцінки загальної якості роботи архітектури введемо інтегральний коефіцієнт ефективності η , який визначається як імовірність того, що аномалія буде не лише виявлена, а й правильно класифікована та доставлена аналітику без втрат у чергах:

$$\eta = \prod_{i=1}^n \epsilon_i = (1 - P_{loss}) \cdot (1 - P_{f_neg}) \cdot (1 - P_{noise}) \quad (3.31)$$

де:

- $(1 - P_{loss})$ - ефективність збору (відсутність переповнення буфера);

- $(1 - P_{f_neg})$ - чутливість гібридного методу (мінімізація пропущених атак);
- $(1 - P_{noise})$ - ефективність фільтрації (мінімізація хибних спрацювань за рахунок декомпозиції сезонності).

Висновки до розділу 3

У третьому розділі було розроблено комплексну архітектуру системи виявлення аномалій, що базується на гібридному підході та багаторівневій обробці даних. Проведене математичне моделювання та проектування дозволили дійти наступних висновків:

- Математичне обґрунтування стійкості збору даних: на основі теорії масового обслуговування (модель $M/M/1/K$) розроблено модель функціонування легковагових агентів. Впровадження буферизації з обмеженою ємністю K та механізму Back-pressure дозволяє стабілізувати роботу системи при пікових навантаженнях;
- Формалізація процесів нормалізації та збагачення: запропоновано модель трансформації неструктурованих даних у вектори ознак через послідовне застосування синтаксичного аналізу, типізації та контекстного збагачення. Використання фільтрів Блума на цьому етапі дозволяє оптимізувати перевірку за списками репутації та індикаторами компрометації з імовірнісною точністю, що суттєво знижує навантаження на підсистему зберігання та прискорює загальний конвеєр обробки;
- Синтез гібридної моделі детекції: математично формалізовано гібридну аналітичну модель, що поєднує детерміновані кореляційні правила та імовірнісну декомпозицію часових рядів. Такий підхід дозволяє одночасно виявляти відомі типи атак через індикаторні функції та приховані аномалії через аналіз випадкового залишку, що адаптується до сезонних коливань інфраструктури (добових та тижневих циклів);
- Оптимізація інтерфейсу прийняття рішень: розроблено модель динамічної пріоритезації інцидентів, яка базується на логістичній функції ризику, що враховує критичність активів та ступінь аномальності події. Впровадження алгоритму логарифмічної агрегації сповіщень

дозволяє вирішити проблему «втоми від сповіщень», знижуючи інформаційний шум без втрати важливих інцидентів;

- Архітектурна надійність та масштабованість: спроектована розподілена структура зберігання з використанням шардування та реплікації гарантує високу доступність даних згідно з CAP-теоремою. Визначені параметри конфігурації забезпечують константну швидкість пошуку навіть при лінійному зростанні обсягів накопиченої телеметрії;

Запропонована архітектура є теоретичним фундаментом для подальшої практичної реалізації системи, забезпечуючи необхідний баланс між продуктивністю та точністю детекції.

4 РЕАЛІЗАЦІЯ МОДЕЛІ

Результати проектування архітектури, отримані у попередньому розділі, сформували чіткі математичні та структурні вимоги до рівнів збору, зберігання та аналітики. Проте для створення функціонуючого прототипу системи виявлення аномалій необхідно здійснити перехід від абстрактних моделей до конкретних програмно-технічних засобів.

Основним завданням практичної частини роботи є вибір та налаштування технологічного стека, який здатен забезпечити реалізацію моделі гібридної детекції на основі декомпозиції часових рядів, підтримку розподіленого зберігання з логарифмічною складністю пошуку та гарантовану доставку подій. Кінцевою метою є підтвердження ефективності розроблених методів шляхом проведення експериментів на реальних наборах даних у вигляді журналів подій.

4.1 Запропонований стек технологій

Для об'єктивної оцінки можливостей побудови системи було проведено аналіз існуючих рішень за чотирма критичними доменами:

- Місце для зберігання та індексації журналів;
- Мови та бібліотеки для написання моделі;
- Робота з кореляційними правилами;
- Візуалізація та реагування;

Основним викликом при виборі місця для зберігання журналів є забезпечення стабільної швидкості пошуку при лінійному зростанні обсягів даних.

Хоча деякі реляційні бази даних (наприклад TimescaleDB) оптимізовані для часових рядів, вона залишається обмеженою жорсткою схемою даних. У задачі виявлення аномалій структура журналів подій часто змінюється (додаються нові поля збагачення), що в реляційних моделях призводить до потреби у трудомістких операціях ALTER TABLE та деградації продуктивності індексів.

NoSQL рішення демонструють надзвичайну швидкість запису, проте їхні можливості щодо повнотекстового пошуку за довільними масками (наприклад, пошук фрагмента шкідливого PowerShell-скрипта в середині текстового поля) є недостатніми без використання додаткових пошукових рушіїв.

Реалізація математичної моделі декомпозиції $T(t) + S(t) + e(t)$ вимагає потужного статистичного апарату. Python є де-факто стандартом в цій галузі завдяки бібліотекам *Statsmodels* та *Prophet*. Вони дозволяють з високою точністю виокремлювати сезонність, однак, при спробі побудови системи реального часу на їх основі виникає проблема затримки: дані потрібно постійно експортувати зі сховища, передавати через мережу до аналітичного модуля, обробляти та повертати результат назад. Для високонавантажених систем такий підхід потребує створення складних брокерів повідомлень (наприклад, Apache Kafka) та засобів оркестрації.

Так як розроблена система виявлення аномалій заснована на гібридному підході, важливим компонентом є двигун кореляції. Основним стандартом серед таких двигунів є Sigma правила. Це універсальний формат, який дозволяє описувати логіку виявлення загроз незалежно від платформи. Проте самі по собі файли Sigma є лише текстовими описами. Для їх роботи необхідний бекенд, який у реальному часі буде транслювати ці правила у специфічні запити до обраної бази даних. Самостійна розробка та підтримка такого транслятора для гібридної системи є ресурсомістким завданням.

Кінцевим результатом роботи системи є сповіщення про небезпеку та візуалізації для кращого розуміння контексту аналітиком. Потужним інструментом для графіків є Grafana, але він має обмежені можливості щодо управління життєвим циклом інцидентів (Case Management). Для реагування інцидент недостатньо просто побачити пік на графіку; необхідно мати можливість дослідити детально журнали подій, додати коментарі та змінити статус інциденту.

Проведений аналіз демонструє проблему технологічного розриву: поєднання окремого сховища, аналітичних скриптів на Python та засобів візуалізації Grafana вимагає створення великої кількості «проміжного» коду. Це підвищує ризик

виникнення помилок при передачі даних та ускладнює підтримку цілісності системи.

Виходячи з цього, найбільш доцільним було визначено використання **Elastic Stack (ELK)** як інтегрованого середовища. Стек складається з трьох ключових компонентів [30]:

- **Elasticsearch:** високомасштабований рушій для повнотекстового пошуку та аналітики. Завдяки інвертованому індексу та формату JSON, Elasticsearch забезпечує швидкість обробки, необхідну для виявлення аномального залишку (e) у реальному часі;
- **Logstash:** інструмент збору та обробки даних. Його роль критична для нормалізації та збагачення подій, оскільки він дозволяє трансформувати неструктуровані логи веб-сервера у вектори ознак;
- **Kibana:** платформа для візуалізації та управління інцидентами. Вона дозволяє наочно представити результати аналізу, зіставляючи сезонні коливання(S) та виявлені аномалії.



Рисунок 4.1 - Схема роботи Elastic Stack

ELK вирішує відразу декілька завдань, які були поставлені при проектуванні моделі виявлення аномалій:

- Elasticsearch виконує роль і сховища, і двигуна пошуку, реалізуючи логарифмічну складність доступу до сегментів;
- Модуль Elastic ML дозволяє виконувати імовірнісну декомпозицію безпосередньо в місці зберігання даних. Це знімає проблему передачі великих

масивів даних через мережу, реалізуючи принцип «обробка ближче до даних».

- Платформа має вбудовану підтримку кореляційних правил, що дозволяє реалізувати детерміновану частину гібридної моделі без додаткових інструментів;
- Надає спеціалізований інтерфейс для створення візуалізацій та перегляд алертів, що покращує ефективність роботи аналітиків SOC.

Таким чином, вибір Elastic Stack дозволяє зосередитись на налаштуванні параметрів моделі декомпозиції та правил кореляції, забезпечуючи при цьому промислову надійність та швидкість обробки подій, що відповідає вимогам, викладеним у Розділі 3.

4.2 Конфігурація середовища

Для перевірки ефективності розроблених математичних моделей та гібридного методу детекції було розгорнуто лабораторний стенд на базі платформи віртуалізації. Архітектура стенда імітує типовий сегмент корпоративної мережі, що складається з об'єкта захисту, системи моніторингу та вузла імітації загроз.

До складу лабораторної бази увійшли наступні компоненти:

- Сервер моніторингу та аналітики (SIEM Node): Elastic Stack (Elasticsearch, Kibana, Logstash, Fleet Server).
- Веб-сервер (Target Node): об'єкт дослідження, що генерує потік подій. На хості встановлено агент Elastic Agent, який реалізує модель збору даних з механізмом Back-pressure;
- Вузол імітації загроз (Attacker Node): генерація аномальної активності для перевірки здатності системи виявляти відхилення від нормального профілю роботи сервера.

Для проведення експериментів на веб-сервері було розгорнуто спеціалізований веб-додаток, що містить критичну вразливість типу **Unrestricted File Upload**

(необмежене завантаження файлів). Дана вразливість була обрана як еталонна для тестування гібридної детекції, оскільки вона створює характерний часовий відбиток: послідовність легітимних запитів до форми завантаження, що змінюється аномальною активністю при зверненні до завантаженого веб-шелла.

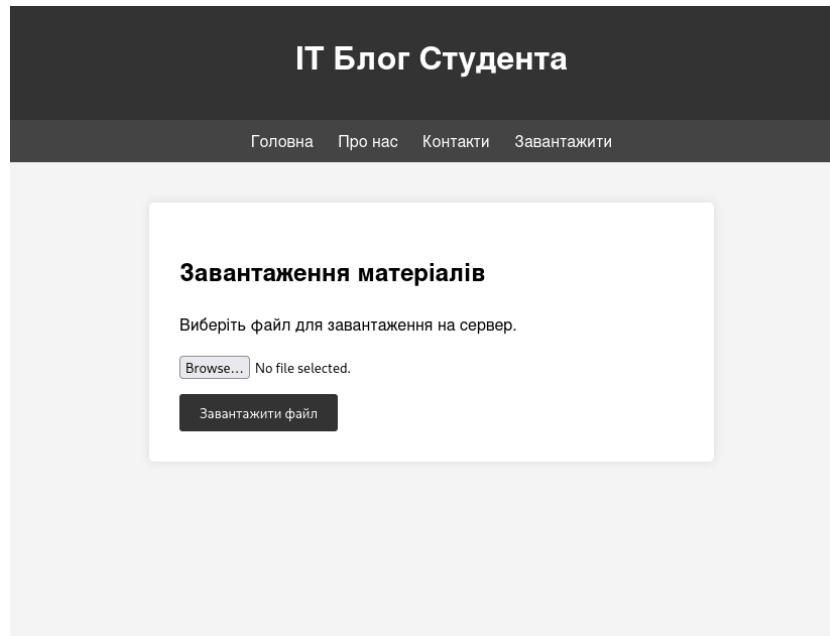


Рисунок 4.2 - Сторінка з вразливістю

Для коректного навчання імовірнісної моделі та формування статистичного профілю нормальної роботи веб-сервера було розроблено спеціалізований скрипт імітації користувацької активності. Використання статичного навантаження не дозволило б перевірити стійкість алгоритму до варіативності даних, тому скрипт реалізує стохастичний підхід до генерації запитів.

Для кожного запиту скрипт обирає випадкову IP-адресу з попередньо визначеного пулу та випадковий рядок User-Agent, що імітує різні браузері та пристрої. Це дозволяє наповнити рівень нормалізації різноманітними векторами ознак. При цьому, скрипт звертається до різних розділів сайту (головна сторінка, конкретні статі, контакти) з випадковими інтервалами затримки, що створює природний "шум" у часовому ряді.

Для перевірки здатності системи фіксувати несанкціоноване проникнення було реалізовано сценарій прямої експлуатації вразливості веб-додатка за допомогою веб-шелла. Атака проводилася з вузла Kali Linux і складалася з наступних кроків:

- Було підготовлено файл `legit.php`, що містить базову PHP-функцію для виконання системних команд через HTTP-запит;
- Використовуючи форму завантаження файлів на цільовому веб-сервері (Debian), було здійснено завантаження створеного веб-шелла. Оскільки серверна частина додатка не здійснювала належної перевірки розширень файлів, скрипт було успішно збережено в директорії `/var/www/html/uploads/`;
- Після завантаження було здійснено звернення до веб-шелла через браузер для виконання команди перевірки поточної директорії та прав користувача.

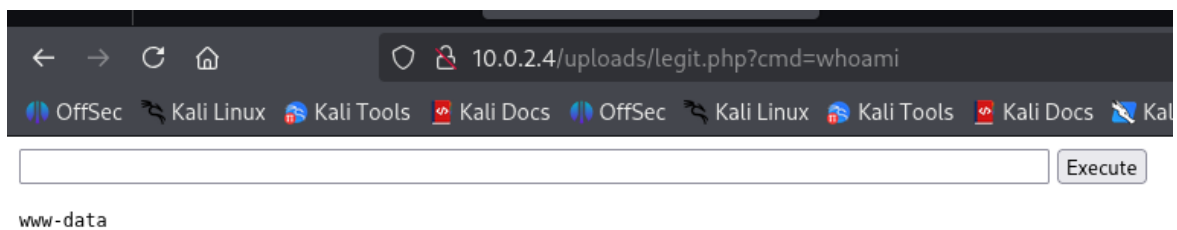


Рисунок 4.3 - Веб-шел у веб-додатку

4.3 Симуляції нормальної роботи

Для отримання об'єктивних результатів детекції експеримент було проведено в умовах, що максимально наближені до реальної експлуатації веб-ресурсу. Тестування складалося з паралельного виконання двох процесів:

- На початку експерименту було активовано розроблений скрипт симуляції нормальної роботи, який безперервно генерував потік HTTP-запитів до веб-сервера. Протягом усього часу тестування скрипт імітував поведінку декількох десятків унікальних користувачів, що зверталися до різних розділів сайту з випадковими інтервалами. Це дозволило сформувати у системі Elastic Stack стабільний часовий ряд нормальної активності;

- Під час активної фази роботи скрипта симуляції, з вузла Kali Linux було здійснено спробу компрометації системи. Не зупиняючи фоновий трафік, було виконано завантаження файлу веб-шелла через вразливу форму. Після успішного завантаження було ініційовано серію запитів до веб-шелла для виконання системних команд. Важливим аспектом цього етапу було те, що шкідливі запити змішувалися з легітимними запитами від скрипта.

4.4 Конфігурація моделі виявлення аномалій

Для практичної реалізації гібридного методу детекції, було налаштоване спеціалізоване завдання для виявлення аномалій (Anomaly Detection Job) у середовищі Elastic Machine Learning. Основною метою налаштування було забезпечення високої чутливості системи до короткочасних аномалій на фоні динамічного трафіку, саме тому модель навчалась на симуляції

Для тестування обраного підходу до виявлення аномалій було встановлено наступні критичні параметри моделі:

- Rare як функція детекції: це ключове налаштування визначає імовірнісну логіку моделі. На відміну від функції count, яка реагує лише на обсяг запитів, функція rare фокусується на виявленні статистично малоімовірних значень у часовому ряді. В контексті атаки це дозволяє ідентифікувати появу веб-шелла, оскільки це ім'я файлу в запитах ні разу не зустрічалося до цього;
- Часове вікно аналізу (Bucket span): Bucket span - це часовий інтервал, протягом якого система агрегує дані перед тим, як передати їх до аналітичної моделі. Вибір короткого інтервалу у 30 секунд зумовлений короткою тривалістю тестування системи скриптом (всього дві години);
- Впливові фактори (Influencers): даний параметр вказує системі, яке поле є першопричиною аномалії. При виявленні відхилення модель не просто генерує сповіщення, а чітко вказує на конкретний шлях до

файлу. Це дозволяє аналітику SOC миттєво локалізувати шкідливий скрипт у директорії /uploads/. Так як метою є виявлення аномальних запитів, було обрано url.path яке зберігає інформацію про шлях до сторінки, до якої був запит;

- Обмеження пам'яті моделі (Model memory limit): оскільки модель працює з текстовими даними (URL-шляхами) у невеликому сегменті мережі, встановлений ліміт у 11 МБ є оптимальним. Це підтверджує "легковаговість" запропонованої архітектури, яка не потребує значних обчислювальних ресурсів для підтримки актуального профілю нормальної поведінки.

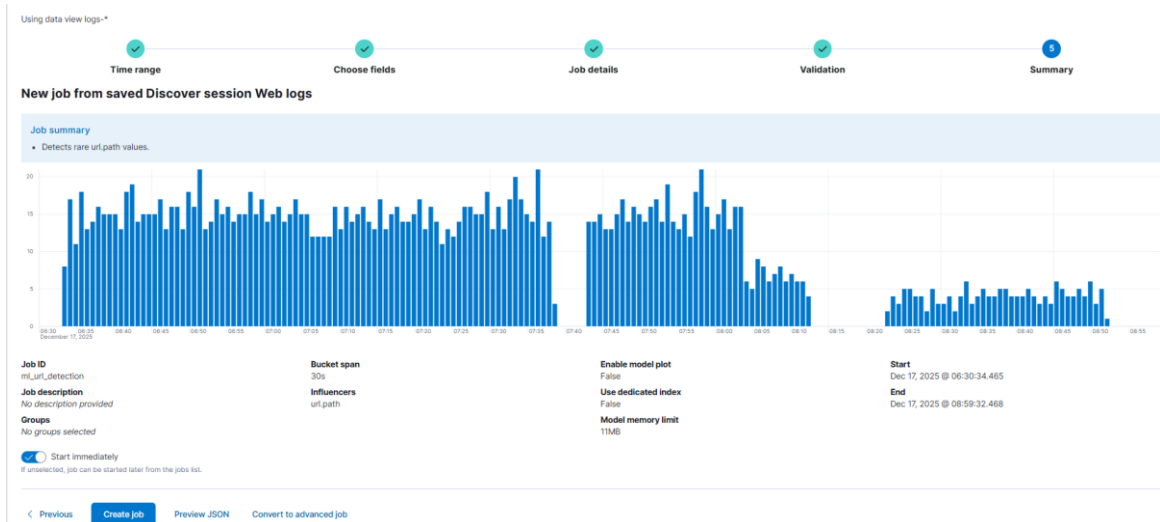


Рисунок 4.4 - Налаштування моделі

4.5 Результати роботи

На виході створюється статистичний еталон «норми». Модель аналізує фоновий трафік та вивчає, які URL-адреси є звичайними для цього сервера. Це дозволяє системі в майбутньому автоматично ігнорувати легітимну активність і фокусуватися лише на аномаліях. В графічному представленні результатом є терактивна часова шкала (Timeline):

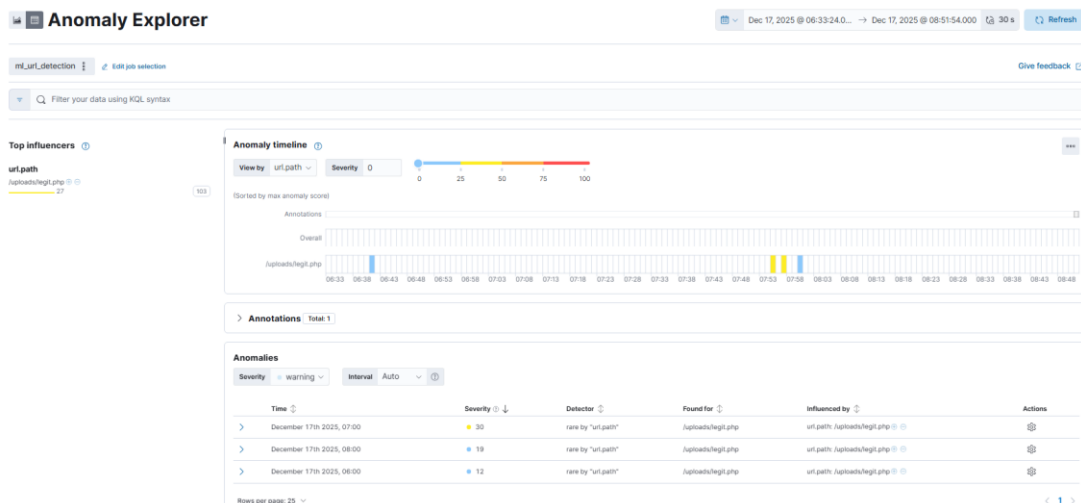


Рисунок 4.5 - Результат роботи моделі

В результаті роботи моделі було отримано:

- Аналітична шкала «Anomaly Timeline»: чіткі часові вікна, у яких відбувалася активність веб-шелла;
- Маркери: жовті маркери вказують на найбільш значущі аномалії, а блакитні маркери на менш критичні відхилення. Це результат звернення до файлу /uploads/legit.php, який був завантажений через вразливість;
- Визначення впливових факторів (Top Influencers): модель не просто сигналізує про аномалію, а видає готовий висновок: «Подія аномальна, тому що цей конкретний URL-шлях є рідкісним для даної системи»;
- Таблиця деталізації «Anomalies»: це фактично готова інформація для реагування, який містить усі необхідні дані для блокування загрози.

Висновки до розділу 4

У четвертому розділі було здійснено практичну реалізацію та експериментальну перевірку спроектованої системи виявлення аномалій. За результатами проведеної роботи сформульовано наступні висновки:

1. На основі порівняльного аналізу існуючих інструментів було підтверджено доцільність використання Elastic Stack. Обрана платформа забезпечує наскрізну інтеграцію рівнів збору, індексації та інтелектуального аналізу, що дозволяє уникнути надлишкової складності при поєднанні розрізнених компонентів;
2. Створено лабораторний стенд, що імітує типову корпоративну мережу, включаючи сервер моніторингу, вразливий веб-сервер та вузол атакуючого. Для формування статистично достовірної моделі "норми" було впроваджено скрипт імітації легітимного трафіку, що генерував випадкові HTTP-запити з рандомізацією IP-адрес та клієнтських додатків;
3. Сконфігуровано інтелектуальне завдання детекції аномалій з використанням імовірнісної функції rare. Таке налаштування дозволило системі вийти на робочий режим Online Learning та сформувати актуальний профіль сезонності у режимі реального часу;
4. Проведено успішне тестування системи на сценарії експлуатації вразливості Unrestricted File Upload. Модель продемонструвала здатність ідентифікувати діяльність веб-шелла на фоні інтенсивної легітимної активності.

ВИСНОВКИ

У магістерській дисертації розв'язано актуальну науково-практичну задачу підвищення ефективності виявлення зловмисної активності в журналах подій шляхом розробки та впровадження гібридної моделі машинного навчання. За результатами проведеного дослідження сформульовано наступні висновки:

1. Проведено детальний огляд нових технік атак, зокрема складних методів експлуатації веб-уразливостей та впровадження веб-шеллів, які маскуються під легітимні запити. Встановлено, що сучасні зловмисники дедалі частіше використовують нестандартні вектори вторгнення, що робить статичні сигнатурні бази неактуальними та вимагає впровадження нових методик виявлення атак;
2. Математично обґрунтовано доцільність застосування гібридної моделі, яка поєднує детерміновані кореляційні правила та імовірнісну декомпозицію часових рядів;
3. На основі системного аналізу об'єкта дослідження проведено математичний синтез моделі детекції;
4. Спроектовано та впроваджено розподілену систему на базі Elastic Stack. Використання механізмів шардування, реплікації та логарифмічного злиття сегментів забезпечило створення рівня зберігання, що здатний обробляти великі масиви неструктурованих журналів подій у реальному часі без втрати продуктивності.
5. Практичні випробування на базі розгорнутого стенда з використанням сценарію експлуатації вразливості File Upload підтвердили працездатність розробленої моделі. Вбудований модуль Machine Learning успішно ідентифікував аномальне звернення до веб-шелла на фоні інтенсивної симуляції нормального трафіку.

Запропонований підхід дозволяє автоматизувати виявлення нових типів атак у режимі реального часу. На відміну від існуючих рішень, розроблена модель

мінімізує рівень хибних спрацювань за рахунок адаптивного врахування сезонних коливань інфраструктури, що забезпечує надійний захист інформаційних систем від цілеспрямованих зловмисних дій. Таким чином, при впровадженні в реальне середовище (SOC) запропоноване рішення дозволяє автоматизувати виявлення рідкісних та нетипових подій у журналах подій, що значно знижує операційне навантаження на аналітиків безпеки.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. **Verizon Business.** 2024 Data Breach Investigations Report. - 2024. - 101 p. - URL: <https://www.verizon.com/business/resources/reports/dbir/>
2. **ENISA.** ENISA Threat Landscape 2024: July 2023 to June 2024. - European Union Agency for Cybersecurity, 2024. - URL: <https://www.enisa.europa.eu/publications/enisa-threat-landscape-2024>
3. **LOLBAS Project.** Living Off the Land Binaries, Scripts and Libraries (LOLBAS). - URL: <https://lolbas-project.github.io/>
4. **CISA, NSA, FBI.** Joint Cybersecurity Advisory: PRC State-Sponsored Cyber Actor Living off the Land to Evade Detection (AA23-144A). - 2023. - URL: <https://www.cisa.gov/news-events/cybersecurity-advisories/aa23-144a>
5. **CrowdStrike.** 2024 Global Threat Report. - CrowdStrike Holdings, Inc., 2024. - URL: <https://www.crowdstrike.com/global-threat-report/>
6. **Wardle P.** The Art of Memory Forensics: Detecting Fileless Malware in Modern Operating Systems. - No Starch Press, 2023. - 456 p.
7. **Sikorski M., Honig A.** Practical Malware Analysis: The Hands-On Guide to Dissecting Malicious Software. - San Francisco : No Starch Press, 2012. - 800 p.
8. **Monnappa K. A.** Learning Malware Analysis: Explore the concepts of code analysis and reverse engineering. - Packt Publishing, 2018. - 450 p.
9. **ENISA.** Understanding the increase in Supply Chain attacks. - European Union Agency for Cybersecurity, 2021. - URL: <https://www.enisa.europa.eu/publications/supply-chain-attacks>
10. **Birsan Alex.** Dependency Confusion: How I Hacked Into Apple, Microsoft and Dozens of Other Companies. - Medium, 2021. - URL: <https://medium.com/@alex.birsan/dependency-confusion-4a5d2ae38c4>
11. **Taylor A., et al.** Typosquatting in Programming Language Package Managers. - Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security, 2020. - P. 1320-1335.

- 12.**Bilge L., Dumitras T.** Before we knew it: an empirical study of zero-day attacks in the real world. - Proceedings of the 2012 ACM conference on Computer and communications security, 2012. - P. 833-844.
- 13.**Roth F.** How to Write SIGMA Rules. - Nextron Systems, 2018. - URL: <https://www.nextron-systems.com/2018/02/10/write-sigma-rules/>
- 14.**Muniz J., et al.** Security Operations Center: Build, Operate, and Maintain your SOC. - Cisco Press, 2015. - 352 p.
- 15.**Géron A.** Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow. - O'Reilly Media, 2019. - 850 p.
- 16.**Chollet F.** Deep Learning with Python. - Manning Publications, 2017. - 504 p.
- 17.**Frost J.** Confusion Matrix: Definition and Examples. - Statistics By Jim, 2023. - URL: <https://statisticsbyjim.com/glossary/confusion-matrix/>
- 18.**Powers D. M. W.** Evaluation: From Precision, Recall and F-Measure to ROC, Informedness, Markedness and Correlation. - Journal of Machine Learning Technologies, 2011. - Vol. 2, No. 1. - P. 37-63.
- 19.**GeeksforGeeks.** AUC-ROC Curve in Machine Learning. - 2024. - URL: <https://www.geeksforgeeks.org/auc-roc-curve-in-machine-learning/>
- 20.**Ahmed M., Mahmood A. N., Hu J.** A survey of network anomaly detection techniques. - Journal of Network and Computer Applications, 2016. - Vol. 60. - P. 19-31.
- 21.**Brauckhoff D., et al.** Impact of Packet Sampling on Anomaly Detection Metrics. - Proceedings of the 6th ACM SIGCOMM conference on Internet measurement, 2006. - P. 159-164.
- 22.**Ye J., Vilbert S., Chen Q.** An anomaly detection technique based on a chi-square statistic for detecting intrusions into information systems. - Quality and Reliability Engineering International, 2003. - Vol. 19, No. 5. - P. 429-443.
- 23.**Bottou L., Lin C. J.** Support Vector Machine Solvers. - Large Scale Machine Learning, 2007. - Vol. 3. - P. 301-320.

- 24.**Stoian N.-A.** Comparison of Selected Machine Learning Techniques for Cyberattack Anomaly Detection. - Bucharest : University Politehnica of Bucharest, 2020.
- 25.**Chen T., Guestrin C.** XGBoost: A Scalable Tree Boosting System. - Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2016. - P. 785-794.
- 26.**Subba B., Biswas S., Karmakar S.** Unsupervised Algorithms to Detect Zero-Day Attacks: Strategy and Application. - IEEE Access, 2021. - Vol. 9. - P. 1125-1142.
- 27.**Zhiqiang L., et al.** A Survey of Outlier Detection Methodologies. - Artificial Intelligence Review, 2004. - Vol. 14. - P. 127-153.
- 28.**Krzysztoń M., et al.** Network Anomaly Detection Using Deep Learning Autoencoders. - Applied Sciences, 2021. - Vol. 11, No. 13.
- 29.**Gartner.** Market Guide for User and Entity Behavior Analytics. - Gartner Research, 2023. - URL: <https://www.gartner.com/en/documents/4014902>
30. **GeeksforGeeks.** What is Elastic Stack and Elasticsearch? - 2024. - URL: <https://www.geeksforgeeks.org/software-engineering/what-is-elastic-stack-and-elasticsearch/>