МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Чорноморський національний університет імені Петра Могили

Факультет комп'ютерних наук

Кафедра комп'ютерної інженерії

ДОПУЩЕНО ДО ЗАХИСТУ

Завідувачка кафедри,
д-р техн. наук, проф.

_____Ірина ЖУРАВСЬКА

« __ » _____ 202__ р.

КВАЛІФІКАЦІЙНА РОБОТА

НА ЗДОБУТТЯ ОСВІТНЬОГО СТУПЕНЯ БАКАЛАВРА

Система моніторингу внутрішніх вразливостей локальної мережі на основі аналізу шаблонної активності зловмисників

Спеціальність 123 Комп'ютерна інженерія

Освітня програма «Комп'ютерна інженерія»

Здобувач _____ Віктор ВОЛКОВ
підпис

«__» _____ 20__ р.

Керівник д-р техн. наук, професор,
завідувач каф. комп'ютерної інженерії _____ Ірина ЖУРАВСЬКА
підпис
«__» _____ 20__ р.

Миколаїв – 2026

Факультет	Комп’ютерних наук
Кафедра	Комп’ютерної інженерії
Рівень вищої освіти	Перший (бакалаврський)
Освітній ступень	Бакалавр
Спеціальність	123 Комп’ютерна інженерія
Освітня програма	Комп’ютерна інженерія

ЗАТВЕРДЖУЮ
Завідувач кафедри комп’ютерної інженерії
_____ Ірина ЖУРАВСЬКА
«_____» _____ 202__ р.

ЗАВДАННЯ
на кваліфікаційну роботу здобувача
Волкова Віктора Володимировича
(прізвище, ім’я, по батькові здобувача)

1. Тема кваліфікаційної роботи

Система моніторингу внутрішніх вразливостей локальної мережі на основі аналізу шаблонної активності зловмисників

Затверджена наказом по ЧНУ ім. Петра Могили від 25.11.2025 № 294.

2. Строк представлення кваліфікаційної роботи «____» _____ 20__ р.

3. Очікуваний результат роботи та початкові дані, якщо такі потрібні

Очікуваним результатом роботи є розроблення прототипу системи моніторингу внутрішніх вразливостей локальної мережі, орієнтованої на поведінковий аналіз шаблонної активності зловмисників. Система має забезпечувати збір і кореляцію телеметрії з різних джерел локальної інфраструктури, виявлення аномальної мережевої активності та формування інцидентів безпеки. У межах роботи передбачається реалізація взаємодії між центральним компонентом системи та агентом моніторингу. Початковими даними для розробки є результати аналізу сучасних засобів моніторингу, типові джерела мережевої телеметрії, поведінкові патерни зловмисників та вимоги до безпечного автоматизованого реагування на інциденти.

4. Перелік питань, що підлягають розробці:

1) дослідити сучасні засоби діагностики, захисту та моніторингу локальних мереж;

2) обґрунтувати доцільність обраних підходів до реалізації апаратно-програмного комплексу;

3) розробити архітектуру системи моніторингу та захисту внутрішніх вразливостей локальної мережі;

4) реалізувати механізми аналізу поведінкових шаблонів та автоматизованого реагування;

5) перевірити реалізацію шляхом проведення серії експериментів на проникнення у межах тестового стенда.

5. Перелік графічних матеріалів

Презентація

6. Консультанти:

Консультант	Кафедра (організація)	Частина роботи

Керівник роботи _____ Ірина ЖУРАВСЬКА
Особистий підпис *Власне ім'я ПРІЗВИЩЕ*

Здобувач _____ Віктор ВОЛКОВ
Особистий підпис *Власне ім'я ПРІЗВИЩЕ*

Дата видачі завдання «_____» _____ 2025 р.

КАЛЕНДАРНИЙ ПЛАН

виконання кваліфікаційної бакалаврської роботи

Тема: Система моніторингу внутрішніх вразливостей локальної мережі на основі аналізу шаблонної активності зловмисників

№	Найменування роботи	Початок	Закінчення	Примітки
1	Розробка та затвердження завдання на виконання КБР	01.12.2025	07.12.2025	Виконано
2	Огляд літератури за темою роботи	08.12.2025	10.01.2026	Виконано
3	Складання календарного плану КБР	11.1.2026	12.01.2026	Виконано
4	Аналіз предметної області	12.01.2026	20.01.2026	Виконано
5	Розробка проєктних рішень	21.01.2026	05.02.2026	Виконано
6	Моделювання та розробка АПЗ	05.02.2026	11.05.2026	Виконано
7	Перевірка працездатності, тестування та апробація розробленого АПЗ, аналіз результатів тестування	05.02.2026	20.05.2026	Виконано
8	Оформлення КБР та презентації	13.05.2026	22.05.2026	Виконано
9	Перший попередній захист КБР	22.05.2026	22.05.2026	Виконано
10	Другий попередній захист КБР	05.06.2026	05.06.2026	Виконано
11	Завершення оформлення КБР та презентації	01.06.2026	02.06.2026	Виконано
12	Перевірка на академічний плагіат, фальсифікацію та списування	03.06.2026	04.06.2026	Виконано
13	Відгук керівника КБР	05.06.2026	06.06.2026	Виконано
14	Подання КБР рецензенту та рецензування КБР	06.06.2026	06.06.2026	Виконано
15	Подання КБР, її електронної копії та інших документів (відгуку, рецензії)	15.06.2026	17.06.2026	Виконано
16	Захист КБР	24.06.2026	24.06.2026	Виконано

Керівник роботи _____ Ірина ЖУРАВСЬКА

Особистий підпис *Власне ім'я ПРІЗВИЩЕ*

Здобувач _____ Віктор ВОЛКОВ

Особистий підпис *Власне ім'я ПРІЗВИЩЕ*

«____» _____ 20__ р.

АНОТАЦІЯ

до кваліфікаційної бакалаврської роботи
«Система моніторингу внутрішніх вразливостей локальної мережі на основі аналізу шаблонної активності зловмисників»
Здобувач: Волков Віктор Володимирович
Керівник: д-р техн. наук, проф, Журавська Ірина Миколаївна

Актуальність роботи зумовлена зростанням складності локальних комп'ютерних мереж, збільшенням кількості внутрішніх сервісів та підвищенням ризику поширення кіберзагроз усередині інфраструктури після первинної компрометації окремого вузла. Традиційні засоби периметрового захисту не завжди забезпечують своєчасне виявлення горизонтального переміщення зловмисника, а використання шифрованого трафіку ускладнює застосування класичного глибокого аналізу пакетів. У зв'язку з цим актуальним є розроблення системи, здатної аналізувати поведінкові шаблони мережевої активності, корелювати події з різних джерел телеметрії та підтримувати автоматизоване реагування на інциденти.

Об'єктом дослідження є процеси моніторингу та виявлення потенційно небезпечної активності у локальних комп'ютерних мережах (ЛКМ).

Предметом дослідження є методи та засоби аналізу мережевої активності, механізми виявлення поведінкових шаблонів зловмисників, а також архітектурні підходи до побудови систем моніторингу внутрішніх вразливостей локальної мережі.

Метою роботи є підвищення ефективності виявлення, локалізації та реагування на внутрішні мережеві інциденти шляхом застосування поведінкового аналізу шаблонної активності зловмисників і розроблення системи моніторингу внутрішніх вразливостей локальної мережі. Для досягнення поставленої мети було досліджено сучасні засоби діагностики, захисту та моніторингу локальних мереж, визначено їх переваги й обмеження, сформовано вимоги до майбутньої системи, розроблено підхід до класифікації джерел телеметрії та реалізовано основні механізми збору, кореляції й аналізу подій. Спроєктовано та розроблено систему агрегації, обробки та реагування на події.

У першому розділі роботи розглянуто предмет моніторингу локальних мереж, основні джерела телеметрії, сучасні засоби діагностики та моніторингу, а також обґрунтовано необхідність використання поведінкового підходу для виявлення внутрішніх загроз. Проведено порівняльний аналіз систем моніторингу, централізованого збору журналів, IDS/IPS-рішень та засобів поведінкового аналізу мережевої активності.

У другому розділі розроблено структуру системи моніторингу внутрішніх вразливостей локальної мережі та описано основні рівні STS-моделі, яка використовується для класифікації джерел телеметрії та локалізації інцидентів. Запропонований підхід дозволяє визначати рівень походження подій, оцінювати потенційний радіус поширення загрози та швидше приймати рішення щодо ізоляції окремих вузлів або сегментів мережі.

У третьому розділі роботи здобувачем наведено результати практичної реалізації та експериментальної перевірки розробленої системи. Описано реалізацію серверної частини, бази даних, агентських компонентів для різних операційних систем, механізмів приймання та обробки телеметрії, формування інцидентів і контрольованого реагування. Проведено перевірку працездатності системи в тестовому середовищі на сценаріях виявлення вертикального та горизонтального сканування, атак методом грубої сили, бокового переміщення зловмисника, аномального внутрішнього трафіку. Отримані результати підтвердили працездатність розроблених механізмів обробки телеметрії, поведінкового аналізу, локалізації інцидентів і формування дій реагування.

Практична цінність роботи полягає у можливості використання запропонованої системи як додаткового рівня контролю внутрішньої мережевої активності в корпоративних інфраструктурах. Система може бути інтегрована з наявними засобами моніторингу та мережевими пристроями, забезпечуючи збір телеметрії, виявлення аномальної активності, формування інцидентів та підтримку безпечного автоматизованого реагування.

Апробація результатів роботи здійснювалася на XXIII Міжнародній науковій конференції «Ольвійський форум-2026: Стратегії країн Причорноморського регіону в геополітичному просторі» (Миколаїв, 2026), XVII Міжнародній науково-практичній конференції «Free and Open Source Software» (Харків, 2026) та конференції «DevOps fwdays’26» (Київ, 2026).

Кваліфікаційна робота містить 75 с., 9 табл., 24 рис., 1 додаток та 27 джерел посилання.

Ключові слова: *локальна мережа, моніторинг, внутрішні вразливості, телеметрія, поведінковий аналіз, STS-модель, кіберінцидент, автоматизоване реагування, IDS, NDR.*

ABSTRACT

of the Bachelor's Thesis

"Monitoring System for Internal Vulnerabilities of a Local Network Based on the Analysis of Attackers' Patterned Activity"

Applicant: Viktor Volkov

Supervisor: Doctor of Technical Sciences, Professor, Iryna Zhuravska

The relevance of the thesis is determined by the increasing complexity of local computer networks, the growing number of internal services, and the higher risk of cyber threats spreading within the infrastructure after the initial compromise of a single node. Traditional perimeter protection tools do not always provide timely detection of an attacker's lateral movement, while the use of encrypted traffic complicates the application of classical deep packet inspection. Therefore, the development of a system capable of analyzing behavioural patterns of network activity, correlating events from different telemetry sources, and supporting automated incident response is relevant.

The object of the research is the processes of monitoring and detecting potentially dangerous activity in local computer networks.

The subject of the research is the methods and tools for analyzing network activity, mechanisms for detecting behavioural patterns of attackers, as well as architectural approaches to building monitoring systems for internal vulnerabilities of a local network.

The purpose of the thesis is to improve the efficiency of detection, localization, and response to internal network incidents by applying behavioural analysis of attackers' patterned activity and developing a monitoring system for internal vulnerabilities of a local network. To achieve this purpose, modern tools for diagnostics, protection, and monitoring of local networks were studied, their advantages and limitations were identified, requirements for the future system were formulated, an approach to classifying telemetry sources was developed, and the main mechanisms for collecting, correlating, and analyzing events were implemented. A system for aggregating, processing, and responding to events was designed and developed.

The first chapter examines the subject of local network monitoring, the main sources of telemetry, modern diagnostic and monitoring tools, and substantiates the need to use a behavioural approach for detecting internal threats. A comparative analysis of monitoring systems, centralized log collection platforms, IDS/IPS solutions, and behavioural network activity analysis tools was carried out.

The second chapter develops the structure of the monitoring system for internal vulnerabilities of a local network and describes the main levels of the STS model, which is used to classify telemetry sources and localize incidents. The proposed approach makes it possible to determine the level of event origin, assess the potential radius of threat propagation, and make faster decisions regarding the isolation of individual nodes or network segments.

The third chapter presents the results of the practical implementation and experimental validation of the developed system. It describes the implementation of the server-side components, the database, agent components for different operating systems, telemetry collection and processing mechanisms, incident generation, and controlled response mechanisms. The system was tested in a dedicated test environment using

scenarios involving vertical and horizontal port scanning, brute-force attacks, attacker lateral movement, and anomalous internal network traffic. The obtained results confirmed the operability of the implemented telemetry processing mechanisms, behavioral analysis capabilities, incident localization approach, and response action generation mechanisms.

The practical significance of the thesis lies in the possibility of using the proposed system as an additional layer of control over internal network activity in corporate infrastructures. The system can be integrated with existing monitoring tools and network devices, providing telemetry collection, anomaly detection, incident generation, and support for secure automated response.

The research findings were presented and discussed at the XXIII International Scientific Conference “Olbian Forum 2026: Strategies of the Black Sea Region Countries in the Geopolitical Space” (Mykolaiv, 2026), the XVII International Scientific and Practical Conference “Free and Open Source Software” (Kharkiv, 2026), and the “DevOps fwdays’26” conference (Kyiv,2026).

The thesis consists of 75 pages, 9 tables, 24 figures, 1 appendix, and references to 27 sources.

Keywords: *local network, monitoring, internal vulnerabilities, telemetry, behavioural analysis, STS model, cyber incident, automated response, IDS, NDR.*

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

ЛКМ – локальна комп’ютерна мережа

ПЗ – програмне забезпечення

СКУД – система контролю та управління доступом

ACL – Access Control List

API – Application Programming Interface

C2 – Command and Control

DDoS – Distributed Denial of Service

DHCP – Dynamic Host Configuration Protocol

DNS – Domain Name System

DPI – Deep Packet Inspection

EDR – Endpoint Detection and Response

GUI – Graphical User Interface

IDS – Intrusion Detection System

IIS – Internet Information Services

IOC – Indicators of Compromise

IoT – Internet of Things

IPFIX – Internet Protocol Flow Information Export

IPS – Intrusion Prevention System

NAT – Network Address Translation

NDR – Network Detection and Response

OSI – Open Systems Interconnection

SIEM – Security Information and Event Management

SMB – Server Message Block

SNMP – Simple Network Management Protocol

SOC – Security Operations Center

SQL – Structured Query Language

STSM – Structure-Based Telemetry Source Model for Incident Localization

TLS – Transport Layer Security

UPS – Uninterruptible Power Supply

VLAN – Virtual Local Area Network

VPN – Virtual Private Network

ВСТУП

У сучасних умовах розвитку інформаційних технологій локальні комп’ютерні мережі (ЛКМ) стали невід’ємною частиною функціонування підприємств, державних установ, фінансових організацій та промислових об’єктів. Практично всі критично важливі бізнес-процеси сьогодні залежать від стабільності та безпеки мережевої інфраструктури. Збільшення кількості цифрових сервісів, використання хмарних платформ, технологій віртуалізації та віддаленого доступу призводить до постійного ускладнення архітектури ЛКМ (англ. Local Area Network, LAN). У межах однієї організації можуть одночасно функціонувати фізичні сервери, віртуальні машини, контейнеризовані сервіси, мережеве обладнання різних виробників, а також велика кількість кінцевих пристроїв користувачів.

Паралельно зі зростанням складності інфраструктури збільшується й кількість потенційних векторів атак. Якщо раніше основна увага приділялася захисту периметра мережі від зовнішніх загроз, то сьогодні значна частина інцидентів інформаційної безпеки пов’язана саме з внутрішньою мережею організації. Причиною можуть бути як дії зовнішнього зловмисника після отримання первинного доступу до інфраструктури, так і помилки конфігурації, використання вразливого програмного забезпечення, недостатній контроль доступу або компрометація окремих хостів.

Одним із найбільш показових прикладів останніх років стала масштабна атака із застосуванням програм-вимагачів WannaCry [17] та NotPetya [11]. У межах цих інцидентів шкідливе програмне забезпечення після проникнення до ЛКМ здійснювало автоматизоване поширення між вузлами, використовуючи внутрішні мережеві взаємодії та вразливості операційних систем. Особливо критичним стало те, що традиційні системи периметрового захисту в багатьох випадках не могли ефективно протидіяти подальшому розповсюдженню загрози всередині інфраструктури [9]. Аналогічні проблеми спостерігалися під час атак на корпоративні середовища з використанням технік lateral movement [5], коли

зловмисник після компрометації одного вузла поступово переміщується між іншими хостами мережі, отримуючи доступ до критичних ресурсів [6, 10].

Окрему проблему становить той факт, що сучасний мережевий трафік дедалі частіше використовує шифрування. Через широке застосування TLS та VPN-технологій класичний глибокий аналіз пакетів (англ. Deep Packet Inspection, DPI) стає або технічно складним, або потребує значних обчислювальних ресурсів. Крім того, у деяких випадках повний аналіз трафіку може створювати ризики порушення конфіденційності або суперечити політикам безпеки організації [7]. У результаті виникає необхідність пошуку альтернативних підходів до моніторингу ЛКМ, які дозволяють виявляти потенційно небезпечну активність без аналізу вмісту кожного пакета.

Актуальність роботи полягає у необхідності створення ефективної системи моніторингу внутрішніх вразливостей ЛКМ, здатної своєчасно виявляти аномальну активність, аналізувати поведінкові шаблони мережевої взаємодії та автоматизовано реагувати на потенційні загрози без суттєвого впливу на продуктивність інфраструктури. Запропонований підхід орієнтований на використання поведінкового аналізу та кореляції подій, що дозволяє виявляти ознаки сканування мережі, атаки методом грубої сили (англ. brute-force), аномальної сервісної активності, несанкціонованих спроб доступу та інших типових сценаріїв зловмисної поведінки.

Важливим фактором є також те, що сучасні корпоративні мережі часто мають гетерогенну структуру та включають системи на базі різних операційних платформ, зокрема Windows, Linux та FreeBSD. У подібних середовищах централізований контроль безпеки значно ускладнюється через різні механізми логування, моделі доступу та інструменти адміністрування. Крім того, великі інфраструктури можуть містити сотні або тисячі кінцевих вузлів, що робить використання ресурсоємних засобів моніторингу економічно невигідним або технічно складним. Саме тому одним із ключових напрямів дослідження є побудова масштабованої системи моніторингу з мінімальним навантаженням на кінцеві хости.

Сучасні виклики потребують ретельного дослідження та обробки, що дозволить досягти глибокого розуміння патернів, а разом з цим дозволить підвищити швидкість реагування на інциденти.

Об’єктом дослідження є процеси моніторингу та виявлення потенційно небезпечної активності у локальних комп’ютерних мережах, зокрема процеси мережевої взаємодії вузлів, формування телеметрії та поширення аномальної активності всередині інфраструктури.

Предметом дослідження є методи та засоби аналізу мережевої активності, механізми виявлення поведінкових шаблонів зловмисників, а також архітектурні підходи до побудови систем моніторингу внутрішніх вразливостей локальної мережі.

Метою роботи є підвищення ефективності виявлення, локалізації та реагування на кіберінциденти шляхом розробки системи моніторингу внутрішніх вразливостей локальної мережі на основі аналізу шаблонної активності зловмисників.

Для досягнення поставленої мети необхідно вирішити такі задачі:

1) дослідити сучасні засоби діагностики, захисту та моніторингу локальних мереж;

2) обґрунтувати доцільність обраних підходів до реалізації апаратно-програмного комплексу;

3) розробити архітектуру системи моніторингу та захисту внутрішніх вразливостей локальної мережі;

4) реалізувати механізми аналізу поведінкових шаблонів та автоматизованого реагування;

5) перевірити реалізацію шляхом проведення серії експериментів на проникнення у межах тестового стенда.

Практична цінність роботи полягає у можливості використання запропонованої системи в реальних корпоративних мережах для підвищення рівня контролю внутрішнього трафіку та своєчасного реагування на інциденти

інформаційної безпеки. Система може бути інтегрована в існуючу інфраструктуру без необхідності суттєвих змін мережевої архітектури та використовуватися як додатковий рівень контролю поряд із традиційними міжмережевими екранами та системами захисту периметра.

Запропонований підхід також дозволяє зменшити навантаження на мережеву інфраструктуру та сервери моніторингу завдяки використанню аналізу метаданих мережевих з’єднань замість повного аналізу трафіку. Це робить систему придатною для використання навіть у середовищах з обмеженими обчислювальними ресурсами або великою кількістю кінцевих пристроїв.

Апробація результатів роботи здійснювалася на XXIII Міжнародна наукова конференція «Ольвійський форум – 2026: стратегії країн Причорноморського регіону в геополітичному просторі» (м. Миколаїв, 29 червня – 4 липня 2026 р.) [1]; XVII Міжнародній науково-практичній конференції «Free and Open Source Software» (11 – 12 лютого 2026 р.) [2].

1 АНАЛІЗ МЕТОДІВ ТА СИСТЕМ МОНІТОРИНГУ ВРАЗЛИВОСТЕЙ МЕРЕЖ ТА ІНФРАСТРУКТУРИ

1.1 Предмет моніторингу та джерела телеметрії

Сучасні локальні комп’ютерні мережі (ЛКМ) формують складне багаторівневе середовище, у межах якого одночасно функціонують сервери, робочі станції, мережеве обладнання, системи віртуалізації, периферійні пристрої та сервіси інфраструктури. Кожен із компонентів генерує значний обсяг технічної інформації, що може використовуватись для аналізу стану мережі, виявлення аномалій та локалізації потенційних інцидентів інформаційної безпеки. Сукупність таких даних формує телеметрію мережевої інфраструктури.

Ефективність систем моніторингу безпосередньо залежить від типу телеметрії, джерела її походження, рівня деталізації та можливості кореляції подій між різними компонентами інфраструктури. При цьому окреме джерело даних не дозволяє сформувати повну картину стану мережі, оскільки кожен тип телеметрії має власні обмеження, переваги та контекст використання.

Мережева телеметрія містить інформацію про мережеві взаємодії між вузлами, параметри з’єднань, характеристики сесій та напрямки передачі трафіку. Такий тип даних дозволяє виявляти аномальні шаблони мережевої активності, горизонтальне сканування, нетипові маршрути взаємодії та ознаки латерального руху (англ. lateral movement) усередині ЛКМ.

Логи міжмережевих екранів містять інформацію про дозволені та заблоковані з’єднання, застосовані політики доступу, NAT-перетворення та події фільтрації трафіку. Подібні журнали дозволяють аналізувати спроби несанкціонованого доступу, порушення правил сегментації та нетипові мережеві взаємодії між сегментами інфраструктури.

Системні журнали операційних систем містять події автентифікації, запуску служб, зміни привілеїв, системні помилки та журнали аудиту. Аналіз таких подій дозволяє виявляти ознаки компрометації вузлів, несанкціоновані зміни конфігурації та аномальну поведінку користувачів або сервісів.

Телеметрія кінцевих точок включає інформацію про процеси, файлову активність, мережеві з’єднання, локальні політики безпеки та взаємодію прикладного програмного забезпечення із системою. Такий тип телеметрії забезпечує високий рівень деталізації та дозволяє здійснювати поглиблений аналіз активності окремих вузлів інфраструктури.

Мережеві сервіси та інфраструктурні компоненти, зокрема DNS, DHCP, VPN, проксі-сервіси та служби каталогів, також є важливими джерелами телеметрії. Дані від таких компонентів дозволяють виявляти аномальні маршрути взаємодії, нетипову активність клієнтів та приховані канали мережевої комунікації.

Попри значний обсяг доступної телеметрії, використання окремих джерел даних часто ускладнює процес локалізації інцидентів через відсутність єдиного підходу до структуризації інформації. У результаті навіть за наявності великої кількості журналів та подій визначення джерела проблеми, потенційного вектора атаки та меж поширення інциденту може вимагати значного часу.

Додатковою проблемою сучасних систем моніторингу є фрагментація телеметрії між різними компонентами інфраструктури. Події безпеки, мережеві журнали, системна телеметрія та сервісні метрики часто аналізуються ізольовано одна від одної, що ускладнює формування цілісного контексту інциденту та виявлення взаємозалежностей між окремими подіями.

У більшості випадків існуючі системи моніторингу концентруються на фіксації окремих подій або аномалій, однак не враховують їх положення у структурі інфраструктури. Через це ускладнюється оцінка потенційного радіуса поширення загрози, визначення суміжних сегментів мережі, які можуть бути скомпрометовані, а також швидка локалізація критичних точок взаємодії між компонентами системи.

Окрему складність становить необхідність оперативного реагування на інциденти в умовах динамічної мережевої взаємодії. Поки виконується аналіз журналів, кореляція подій та уточнення джерела аномальної активності, загроза може поширюватися між вузлами ЛКМ, уражаючи суміжні сервіси або критичні

компоненти інфраструктури. У подібних умовах особливого значення набуває структурований підхід до класифікації телеметрії та локалізації інцидентів відповідно до рівня їх походження.

1.2 Аналіз засобів діагностики та моніторингу мереж

Зі збільшенням складності ЛКМ та кількості внутрішніх сервісів постійно зростають вимоги до систем моніторингу та виявлення інцидентів інформаційної безпеки. Сучасна інфраструктура генерує значний обсяг телеметрії, що включає мережеві події, системні журнали, дані прикладних сервісів та інформацію про стан обладнання. У результаті системи моніторингу поступово еволюціонували від простого контролю доступності вузлів до комплексного аналізу поведінкових характеристик мережевої взаємодії.

Важливим аспектом, який часто виходить з поля зору інженерів, особливо за відсутності системного підходу до адміністрування інфраструктури, є системи обліку та інвентаризації ресурсів. Зі збільшенням кількості вузлів, сервісів та периферійних пристроїв використання людської пам’яті як основного механізму інвентаризації стає практично неможливим. У результаті значна кількість підприємств змушена покладатися на неповні або неактуальні дані щодо структури власної мережі, що створює додаткові ризики при виявленні несанкціонованих пристроїв або локалізації інцидентів. Саме тому системи обліку інфраструктури доцільно розглядати як один із базових компонентів сучасного моніторингу.

Подібні рішення, зокрема NetBox (рис. 1.1), GLPI або OCS Inventory, дозволяють централізовано підтримувати інформацію про структуру мережі, адресний простір, конфігурацію обладнання та взаємозв’язки між вузлами [3]. Наявність актуальної інформації про топологію інфраструктури суттєво спрощує адміністрування, сегментацію мережі та локалізацію інцидентів. Водночас подібні системи практично не виконують аналізу поведінки вузлів або мережевої взаємодії в реальному часі, через що не можуть використовуватись як самостійний засіб виявлення загроз.

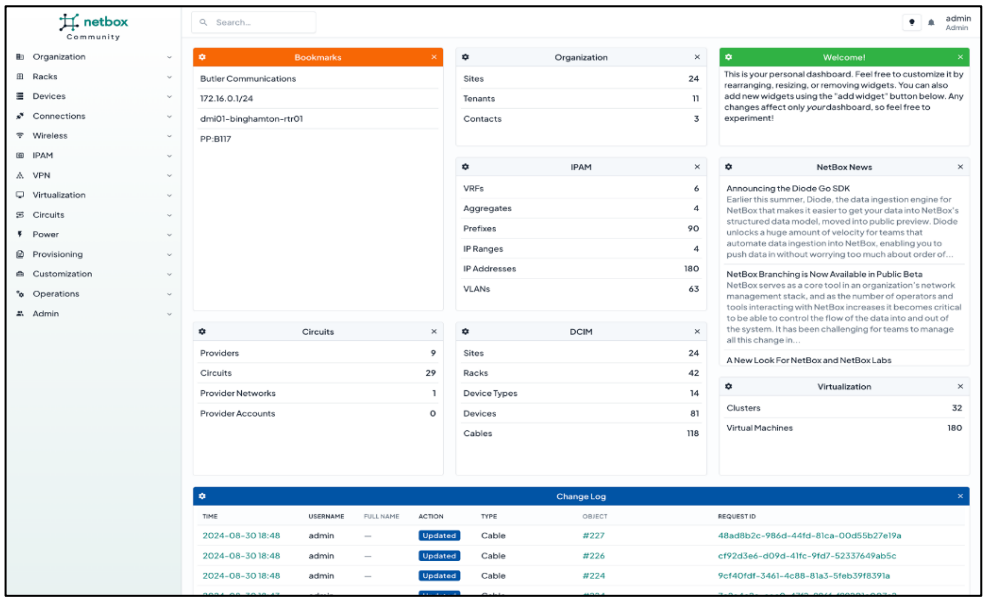


Рисунок 1.1 – Приклад головної сторінки системи NetBox [4]

Наступним етапом розвитку стали системи класичного моніторингу інфраструктури, основною задачею яких є контроль доступності сервісів, навантаження на сервери, стану мережевого обладнання та базових параметрів продуктивності. До найбільш поширених рішень цього класу належать Zabbix, Nagios, Prometheus (рис. 1.2) та PRTG. Подібні системи ефективно виявляють відмови обладнання, деградацію сервісів або перевищення порогових значень навантаження.

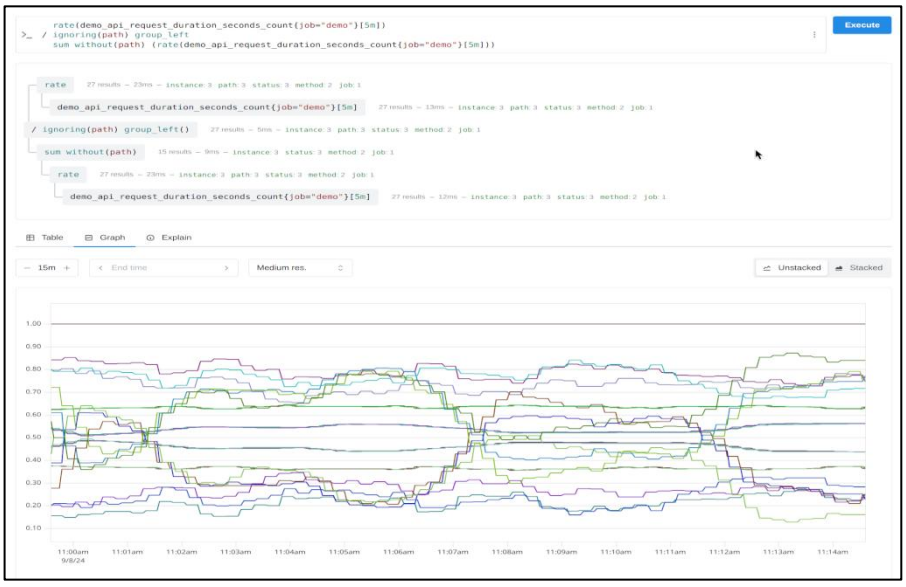


Рисунок 1.2 – Демонстрація можливостей збору, обробки та візуалізації метрик системою Prometheus 3.0 [23]

Разом із цим класичні системи моніторингу мають обмежені можливості щодо аналізу поведінкових характеристик мережевої взаємодії. У більшості випадків вони орієнтовані на контроль окремих метрик та порогових значень, а не на аналіз контексту подій або взаємозв’язків між вузлами мережі. У результаті подібні системи здатні своєчасно виявити перевантаження сервера або втрату доступності сервісу, однак не дозволяють визначати горизонтальне поширення загроз, аномальну взаємодію між сегментами або нетипову мережеву активність.

Подальший розвиток систем моніторингу призвів до широкого використання платформ централізованого збору та кореляції журналів подій. До найбільш поширених рішень цього класу належать Elastic Stack (рис. 1.3), Graylog (рис. 1.4), Splunk та Wazuh [20□22]. Основною задачею подібних платформ є централізований збір логів із різних джерел, їх нормалізація, індексація та подальша кореляція подій. Подібний підхід дозволяє об’єднувати різнорідні джерела телеметрії в єдиному середовищі аналізу та виявляти складні сценарії атак, що охоплюють декілька вузлів або сервісів одночасно.

Рисунок 1.3 – Приклад побудови дашборду системою Elastic Stack на базі зібраних логів [24]

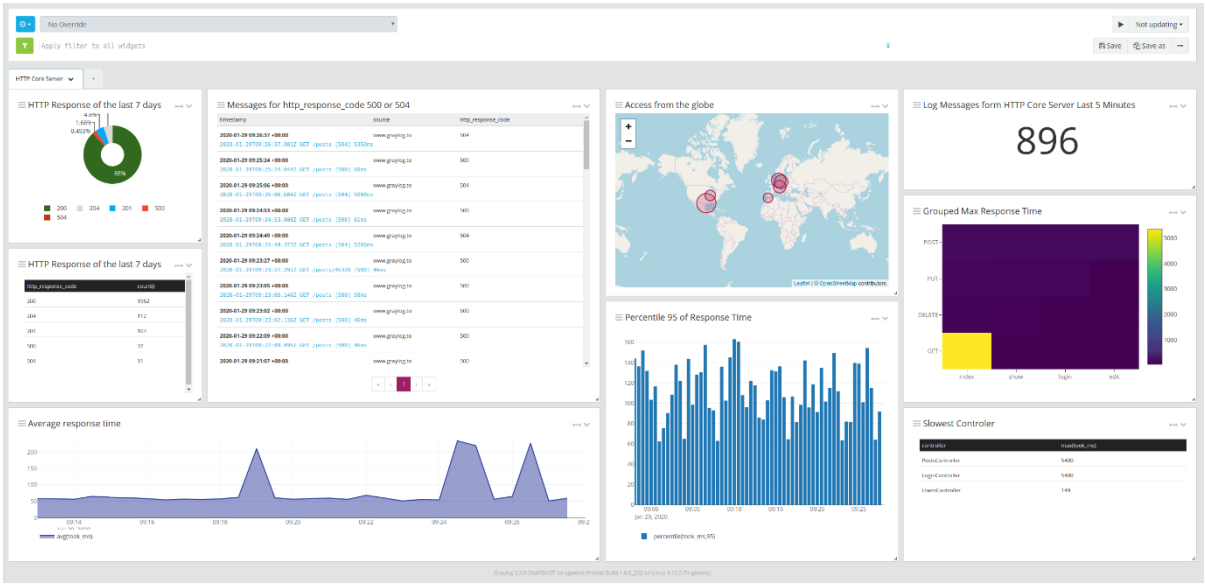


Рисунок 1.4 – Обробка журналу Syslog засобами Graylog [25]

Разом із цим централізований збір логів створює значне навантаження на інфраструктуру зберігання та обробки даних. Значна частина журналів містить велику кількість низькорівневих подій, що ускладнює виділення дійсно важливих індикаторів компрометації. Додатковою проблемою є різнорідність форматів журналів та необхідність постійної нормалізації подій, оскільки різні джерела телеметрії використовують різні структури записів. У результаті зі збільшенням масштабу інфраструктури зростає складність підтримки та кореляції подій, а також кількість хибнопозитивних спрацювань.

Окремий клас рішень формують системи IDS та IPS, орієнтовані на аналіз мережевого трафіку та виявлення атак у режимі реального часу. До найбільш поширених систем цього класу належать Snort та Suricata [18; 19]. Основним принципом їх роботи є сигнатурний аналіз мережевого трафіку та використання правил виявлення відомих шаблонів атак [12]. Подібний підхід дозволяє ефективно виявляти мережеве сканування, спроби експлуатації відомих вразливостей, brute-force атаки та інші типові сценарії мережевих загроз (рис. 1.5).

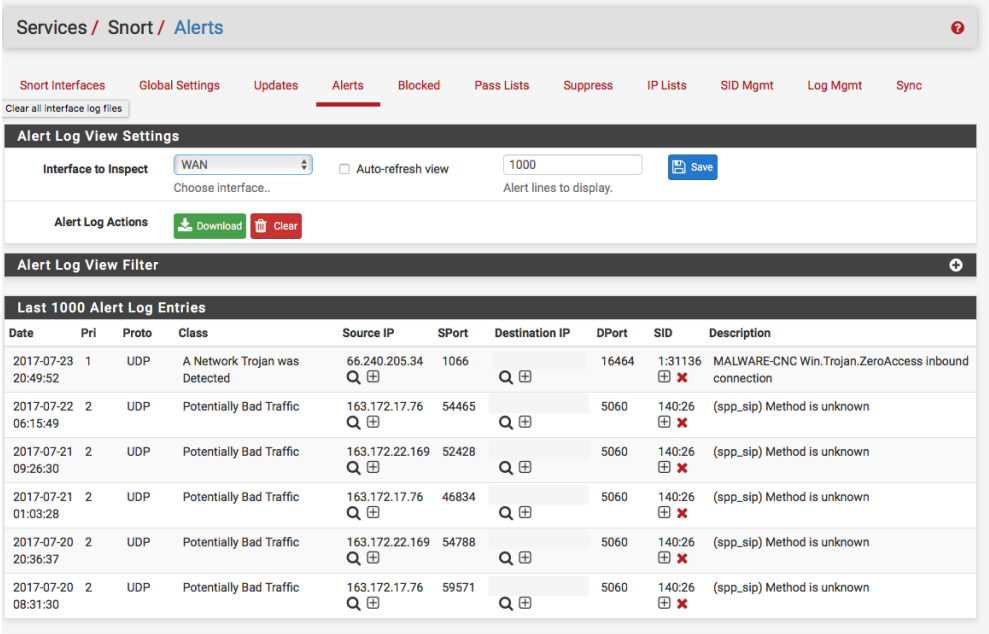


Рисунок 1.5 – Події, знайдені Snort,
вбудованим в програмний маршрутизатор pfSense [26]

Незважаючи на високу ефективність сигнатурного аналізу, його можливості суттєво обмежуються в умовах сучасної інфраструктури. Значне поширення TLS-шифрування, використання контейнеризації, динамічної маршрутизації та мікросервісної архітектури ускладнюють повноцінний аналіз вмісту трафіку [13]. Крім того, системи DPI можуть створювати значне навантаження на мережеву інфраструктуру та вимагати суттєвих обчислювальних ресурсів [8; 15].

Подальший розвиток засобів моніторингу призвів до появи систем поведінкового аналізу та рішень класу Network Detection and Response (NDR). Подібні системи орієнтовані не лише на аналіз окремих пакетів або журналів подій, а й на виявлення аномальних шаблонів мережевої взаємодії між вузлами ЛКМ [16]. Для цього використовуються мережеві метадані, статистичні характеристики сесій, інформація про напрямки взаємодії та поведінкові моделі активності.

На відміну від класичних IDS/IPS-систем, поведінковий підхід дозволяє виявляти аномалії навіть у зашифрованому трафіку без необхідності повного аналізу вмісту пакетів. Особливо ефективним він є для виявлення горизонтального переміщення зловмисника, нетипової сервісної активності, аномальних шаблонів доступу та прихованої взаємодії між сегментами мережі. Разом із цим поведінковий

аналіз потребує якісної кореляції телеметрії та формування базової моделі нормальної активності інфраструктури.

Для глибокого розуміння можливостей та обмеження систем, які активно використовуються в сучасних передових інфраструктурах, було проведене порівняння в табл. 1.1

Таблиця 1.1 – Порівняння інструментів обслуговування мережі

Характеристика	**Системи моніторингу**	**Централізований збір логів / SIEM**	**IDS/IPS**	**Поведінковий аналіз / NDR**
Приклади рішень	Zabbix, Nagios, Prometheus	Elastic Stack, Graylog, Wazuh	Snort, Suricata	Darktrace, Vectra AI
Основний тип даних	Метрики	Логи	Мережевий трафік	Телеметрія та поведінкові патерни
Основна задача	Контроль доступності та навантаження	Кореляція подій	Виявлення атак	Виявлення аномальної активності
Основний підхід	Порогові значення	Аналіз журналів	Сигнатурний аналіз	Аналіз поведінки
Робота із TLS-трафіком	Обмежена	Часткова	Ускладнена	Ефективна
Виявлення невідомих загроз	Обмежене	Часткове	Низьке	Високе
Аналіз Lateral Movement	Практично відсутній	Частковий	Обмежений	Ефективний
Навантаження на інфраструктуру	Низьке	Високе	Високе	Середнє
Масштабованість	Висока	Обмежується обсягом логів	Складна	Висока
Реакція на інциденти	Сповіщення	Сповіщення та кореляція	Блокування окремих подій	Поведінкова локалізація та ізоляція

Таким чином, аналіз сучасних підходів до моніторингу демонструє поступовий перехід від ізольованого контролю окремих вузлів або подій до комплексного поведінкового аналізу мережевої інфраструктури. Класичні системи моніторингу забезпечують високий рівень контролю доступності та стану інфраструктури, однак їх можливості є недостатніми для ефективного виявлення складних внутрішніх загроз та швидкої локалізації інцидентів у межах локальної мережі. Саме тому актуальним залишається створення систем, здатних поєднувати аналіз телеметрії різних рівнів із поведінковим підходом до виявлення аномальної активності.

1.3 Постановка задачі та вимоги до системи

Аналіз сучасних підходів до моніторингу мережевої інфраструктури демонструє, що більшість існуючих рішень орієнтовані або на контроль окремих метрик, або на централізований збір журналів подій, або на сигнатурне виявлення відомих загроз. При цьому сучасна інфраструктура дедалі частіше будується як розподілене середовище, у якому велика кількість сервісів, вузлів та систем безпеки функціонують одночасно та генерують значний обсяг різнорідної телеметрії.

Подібний підхід створює низку практичних проблем. Для підтримки великої кількості спеціалізованих систем необхідні глибокі знання окремих платформ, складна інтеграція між компонентами та постійне супроводження механізмів кореляції подій. У результаті адміністрування інфраструктури поступово перетворюється на підтримку окремих ізольованих сервісів, між якими часто відсутня повноцінна взаємодія.

Одним із суттєвих недоліків сучасних distributed-рішень є обмежені можливості автоматизованого реагування на інциденти. У більшості випадків виявлення загрози одним компонентом системи призводить лише до формування сповіщення або запису журналу подій без подальших активних дій. Особливо критично це проявляється у випадках, коли інцидент вже виходить за межі окремого вузла та зачіпає цілі сегменти локальної мережі.

Додатковою проблемою є орієнтація значної частини сучасних систем на аналіз окремих подій, а не поведінкових характеристик мережевої взаємодії. У результаті системи можуть ефективно виявляти відомі сигнатури атак, однак мають обмежені можливості щодо виявлення нетипової активності, горизонтального переміщення зловмисника або аномальної взаємодії між сегментами мережі. У межах даної роботи ставиться задача розробки концепції системи моніторингу локальної мережі, орієнтованої на поведінковий аналіз мережевої взаємодії, кореляцію телеметрії з різних рівнів інфраструктури та автоматизовану локалізацію потенційних інцидентів. На відміну від класичних систем

моніторингу, запропонований підхід передбачає не лише фіксацію подій або формування сповіщень, а й можливість виконання активних дій у відповідь на виявлену аномальну активність.

Основною задачею системи є виявлення поведінкових аномалій у межах локального сегмента мережі, визначення потенційного радіуса поширення загрози та забезпечення швидкої локалізації інциденту. Особлива увага приділяється виявленню сканування портів, brute-force атак, нетипових маршрутів взаємодії між вузлами, аномальної сервісної активності та ознак горизонтального переміщення між сегментами мережі.

У випадку виявлення критичної аномальної активності система повинна підтримувати механізми автоматизованого реагування. Подібні механізми можуть включати динамічне блокування мережевої взаємодії, ізоляцію окремих вузлів або сегментів, зміну політик доступу та переведення окремих компонентів у контрольований безпечний стан. Для неприорітетних вузлів інфраструктури також може передбачатися можливість примусового обмеження активності або контрольованого відключення від мережевої взаємодії.

З огляду на зазначені задачі до системи висуваються наступні функціональні вимоги:

– збір телеметрії з різних рівнів інфраструктури;
– централізована кореляція подій;
– підтримка поведінкового аналізу мережевої активності;
– виявлення аномальної взаємодії між вузлами;
– локалізація потенційно скомпрометованих сегментів;
– підтримка автоматизованого реагування на інциденти;
– можливість масштабування в межах локальної мережі;
– підтримка інтеграції із зовнішніми засобами моніторингу та мережевими пристроями.

Окрім функціональних вимог, система повинна відповідати низці нефункціональних вимог. Зокрема, важливими є мінімізація впливу на

продуктивність мережевої інфраструктури, забезпечення відмовостійкості, підтримка модульної архітектури та можливість роботи в умовах великого обсягу телеметрії. Додатково система повинна забезпечувати безпечну взаємодію між компонентами та підтримувати сегментований підхід до обробки мережевих подій.

У межах даної роботи основна увага приділяється аналізу мережевої взаємодії всередині локального сегмента мережі та побудові підходу до поведінкової локалізації інцидентів. Практична реалізація окремих механізмів автоматизованого реагування, зокрема багаторівневого підтвердження критичних дій або фізичного обмеження інфраструктурних компонентів, виходить за межі даної роботи та розглядається як напрямок подальшого розвитку системи.

1.4 Задачі моніторингу внутрішніх вразливостей ЛКМ)

У межах даної роботи розглядається задача побудови системи моніторингу локальної мережі, орієнтованої на поведінковий аналіз телеметрії, локалізацію інцидентів інформаційної безпеки та автоматизоване реагування на аномальну активність. Основною особливістю запропонованого підходу є використання структурованої моделі класифікації джерел телеметрії для визначення рівня походження подій, оцінки взаємозв’язків між сегментами інфраструктури та визначення потенційного радіуса поширення загрози.

На відміну від класичних систем моніторингу, які переважно орієнтовані на аналіз окремих журналів подій або контроль порогових значень метрик, у роботі пропонується підхід до централізованої кореляції телеметрії різних рівнів із подальшим аналізом поведінкових характеристик мережевої взаємодії. Подібний підхід дозволяє підвищити швидкість локалізації інцидентів, зменшити залежність від ручного аналізу подій та спростити визначення потенційно скомпрометованих сегментів мережі.

Для досягнення поставленої мети у роботі вирішуються наступні задачі:

– аналіз та класифікація джерел телеметрії локальної мережі відповідно до рівня їх походження в інфраструктурі;

– розробка моделі структурованого представлення телеметрії для локалізації інцидентів та аналізу взаємозв’язків між сегментами мережі;

– реалізація механізму централізованого збору та кореляції мережевих подій із різних джерел телеметрії;

– виявлення поведінкових аномалій, пов’язаних із нетиповою мережевою активністю, скануванням портів, brute-force атаками та аномальною взаємодією між вузлами локальної мережі;

– визначення потенційного радіуса поширення інциденту на основі аналізу мережевої взаємодії та структури сегментів інфраструктури;

– реалізація механізму автоматизованої локалізації потенційно скомпрометованих вузлів або сегментів мережі;

– формування правил реагування на виявлену аномальну активність, включаючи динамічне блокування мережевої взаємодії або ізоляцію окремих сегментів;

– дослідження можливості інтеграції системи з існуючими засобами моніторингу, мережевим обладнанням та системами журналювання подій.

Для оцінки ефективності запропонованого підходу у межах роботи також можуть бути розглянуті тестові сценарії, що моделюють типові інциденти інформаційної безпеки в локальній мережі. До таких сценаріїв можуть належати:

– виявлення сканування портів усередині локального сегмента мережі;

– виявлення brute-force атаки на мережевий сервіс;

– фіксація нетипового збільшення кількості мережевих взаємодій між вузлами;

– виявлення аномального east-west трафіку між сегментами інфраструктури;

– локалізація сегмента мережі, у якому виникла аномальна активність;

– перевірка механізму автоматизованої ізоляції окремого вузла або сегмента;

– аналіз зміни поведінкових характеристик мережі після імітації компрометації одного з вузлів;

– перевірка механізму кореляції подій із різних рівнів телеметрії.

У результаті виконання роботи пропонується підхід до побудови системи моніторингу локальної мережі, здатної поєднувати поведінковий аналіз мережевої взаємодії із структурованою обробкою телеметрії для швидкої локалізації інцидентів та підтримки автоматизованого реагування на загрози.

Висновки до розділу 1

У межах розділу було проведено аналіз сучасних підходів до моніторингу локальних мереж та засобів виявлення інцидентів інформаційної безпеки. Розглянуто основні джерела телеметрії, особливості систем моніторингу інфраструктури, платформ централізованого збору журналів подій, IDS/IPS-рішень та систем поведінкового аналізу мережевої активності.

Проведений аналіз показав, що сучасна мережева інфраструктура генерує значний обсяг різнорідної телеметрії, однак більшість існуючих рішень спеціалізуються на окремих класах задач та потребують складної інтеграції між компонентами. У результаті ускладнюється кореляція подій, локалізація інцидентів та оцінка потенційного радіуса поширення загроз у межах локальної мережі.

Встановлено, що класичні підходи, орієнтовані на сигнатурний аналіз або контроль окремих метрик, мають обмежену ефективність в умовах сучасної інфраструктури, яка характеризується використанням TLS-шифрування, контейнеризації, мікросервісної архітектури та високою динамічністю мережевої взаємодії. У зв’язку з цим доцільно здійснювати поступовий перехід до поведінкового аналізу телеметрії та використання систем класу NDR.

На основі проведеного аналізу сформовано вимоги до системи моніторингу локальної мережі, орієнтованої на поведінковий аналіз, централізовану кореляцію телеметрії та автоматизовану локалізацію інцидентів. Також визначено основні задачі, що мають бути вирішені у межах роботи.

2 РОЗРОБЛЕННЯ СИСТЕМИ МОНІТОРИНГУ ВНУТРІШНІХ ВРАЗЛИВОСТЕЙ МЕРЕЖІ

2.1 Основні рівні STS-моделі

Однією з основних проблем під час реагування на мережеві інциденти є не лише виявлення факту підозрілої активності, а й визначення меж її можливого впливу на інфраструктуру. У багатьох випадках система моніторингу може зафіксувати конкретну подію: невдалу спробу автентифікації, підозріле мережеве з’єднання, спробу сканування портів, зміну стану сервісу або відмову окремого пристрою. Проте сама наявність такої події не завжди дає відповідь на практично важливе питання: які вузли, сервіси, облікові записи, канали доступу або сегменти мережі після цього інциденту тимчасово не можна вважати довіреними.

Для вирішення цієї задачі у роботі використовується авторська структурна модель джерел телеметрії для локалізації інцидентів (Structure-Based Telemetry Source Model STS). Її призначення полягає у класифікації джерел телеметрії відповідно до їх ролі в інфраструктурі, рівня походження подій та значення для визначення зони потенційної компрометації. Модель STS не розглядається як готовий галузевий стандарт або заміна існуючих систем моніторингу, централізованого збору журналів чи засобів виявлення атак. Її роль полягає у створенні додаткового аналітичного шару, який допомагає пов’язати окрему подію з інфраструктурним контекстом.

Основна ідея моделі полягає в тому, що локальна комп’ютерна інфраструктура не є однорідним середовищем. Вона складається з різних за призначенням рівнів: зовнішніх сервісів доступності, прикордонних засобів контролю, внутрішньої мережевої взаємодії, серверів і робочих станцій, а також фізичних або периферійних компонентів інфраструктури. Кожен із цих рівнів формує власний тип телеметрії та має власні характерні ризики. Тому під час локалізації інциденту важливо не лише визначити джерело сигналу, а й зрозуміти, які суміжні сутності можуть бути пов’язані з ним. Формалізований опис рівнів, сутностей та принципів їх віднесення до відповідних рівнів наведено у табл. 2.1.

Таблиця 2.1 – Рівні STS-моделі та приклади сутностей

Назва рівня	**Сутності та приклади**	**Базовий принцип віднесення до рівня**	**Типові або відомі вразливості сутностей рівня**
Shield Dome / «захисний купол»	Зовнішній моніторинг доступності, перевірки стану сервісів, SLA-моніторинг, зовнішні точки контролю, публічні сервіси спостереження	До рівня належать сутності та сигнали, які відображають зовнішній стан інфраструктури та наслідки інциденту для доступності сервісів. Цей рівень не завжди показує джерело атаки, але допомагає оцінити її вплив	DDoS, деградація доступності, відмова зовнішнього сервісу, помилки DNS, недоступність публічного endpoint,
Border / «кордон»	Міжмережеві екрани, маршрутизатори, шлюзи VPN, проксі-сервери, NAT, ACL, точки міжсегментного контролю	До рівня належать сутності, через які проходить контрольована взаємодія між сегментами мережі або із зовнішніми мережами. Рівень визначає межі доступу та правила проходження трафіку	Компрометація VPN-доступу, використання легітимних облікових записів, помилки ACL, відкриті адміністративні порти, некоректні правила NAT або міжмережевого екрана
Castle / «фортеця»	Локальна мережева взаємодія, VLAN, підмережі, DNS-активність, ARP, мережеві потоки, службові з’єднання, east-west traffic	Рівень описує взаємодію між внутрішніми вузлами інфраструктури. Він дозволяє аналізувати маршрути внутрішнього трафіку, нетипові з’єднання та зміну поведінки всередині локальної мережі	Горизонтальне переміщення зловмисника горизонтальне сканування, ARP spoofing, DNS tunneling, аномальні east-west з’єднання, сканування внутрішніх сервісів
Moat / «рів»	Сервери, робочі станції, віртуальні машини, контейнери, прикладні сервіси, облікові записи, процеси, код, бази даних, файлові ресурси	До рівня належать вузли, сервіси та дані, які безпосередньо обробляють інформацію, виконують бізнес-логіку або можуть бути ціллю для закріплення зловмисника в інфраструктурі	brute-force, EternalBlue, Log4Shell, вразливі вебсервіси, слабкі паролі, підвищення привілеїв, шкідливі процеси, компрометація облікового запису
Ground / «земля»	Системи контролю та управління доступом (СКУД), Uninterruptible Power Supply (UPS), IP-камери, пристрої Internet of Things (IoT), точки доступу, контролери, периферійне обладнання, фізичні канали та живлення	Рівень охоплює фізичну та навколофізичну основу інфраструктури. Через нього можливий вплив на систему шляхом фізичного доступу, компрометації периферійного обладнання або порушення живлення	Вразливі IoT-пристрої, Mirai-подібні атаки, стандартні паролі IP-камер, незахищений Simple Network Management Protocol (SNMP), компрометація Wi-Fi точки, відмова UPS, фізичний доступ до обладнання

У межах моделі STS виділено п’ять логічних рівнів: «захисний купол» (англ. Shield Dome), «кордон» (англ. Border), «фортеця» (англ. Castle), «рів» (англ. Moat) та «земля» (англ. Ground). Назви рівнів використовуються як метафора, що

дозволяє простіше пояснити роль різних частин інфраструктури у процесі локалізації інцидентів.

Логіка моделі побудована навколо того, що до ядра інфраструктури, тобто до сервісів, коду, даних, серверів і робочих станцій, зловмисник зазвичай потрапляє або через мережеву взаємодію, або через фізичний чи навколофізичний рівень інфраструктури. Навіть у випадку використання шкідливого програмного забезпечення, бекдора або викрадених облікових даних, початковий шлях впливу найчастіше пов’язаний із мережею, віддаленим доступом, зовнішнім сервісом, внутрішнім маршрутом взаємодії або фізично доступним обладнанням. Саме тому STS-модель розглядає інцидент не як ізольовану подію, а як сигнал, що може поширювати недовіру на пов’язані з ним сутності.

Рівень Shield Dome / «захисний купол» відображає зовнішній стан інфраструктури. До нього належать метрики доступності, зовнішні перевірки стану сервісів, показники затримок, результати синтетичного моніторингу та інші сигнали, які дозволяють оцінити наслідки інциденту для користувачів або зовнішніх систем. Цей рівень не завжди дозволяє безпосередньо виявити джерело атаки, але він важливий для розуміння впливу інциденту. Наприклад, деградація доступності вебсервісу може бути наслідком DDoS-атаки, перевантаження внутрішнього сервера, помилки маршрутизації або відмови інфраструктурного компонента.

Рівень Border / «кордон» охоплює прикордонні та міжсегментні засоби контролю. До нього належать міжмережеві екрани, маршрутизатори, VPN-шлюзи, проксі-сервери, правила NAT, ACL та інші компоненти, які визначають, які з’єднання дозволені між сегментами або між локальною мережею та зовнішнім середовищем. Події цього рівня мають особливе значення для аналізу віддаленого доступу, спроб обходу політик безпеки, нетипових зовнішніх підключень і порушень міжсегментної взаємодії.

Рівень Castle / «фортеця» описує внутрішню мережеву взаємодію між вузлами. Саме на цьому рівні можуть проявлятися ознаки горизонтального

переміщення зловмисника, внутрішнього сканування, нетипового обміну між сегментами, аномальної DNS-активності або несподіваних службових з’єднань. Для цього рівня важливими є мережеві метадані, напрямки взаємодії, інформація про сесії, кількість унікальних цілей, кількість портів і повторюваність підключень у межах певного часу.

Рівень Moat / «рів» охоплює сервери, робочі станції, віртуальні машини, контейнери, прикладні сервіси, облікові записи, процеси, код і дані. Саме тут розташовані активи, що безпосередньо обробляють інформацію та виконують прикладну логіку. Події цього рівня часто пов’язані з автентифікацією, запуском процесів, доступом до файлів, роботою служб, помилками застосунків або змінами локальної конфігурації. У межах запропонованої моделі цей рівень розглядається як один із ключових для виявлення компрометації окремих вузлів і формування зони недовіри навколо них.

Рівень Ground / «земля» охоплює фізичну та навколофізичну основу інфраструктури. До нього належать СКУД, UPS, IP-камери, IoT-пристрої, точки доступу, периферійне обладнання, контролери та інші пристрої, які часто мають обмежені механізми захисту, але можуть суттєво впливати на роботу всієї інфраструктури. Наприклад, відмова системи резервного живлення або компрометація пристрою відеоспостереження не завжди є класичним мережевим інцидентом у вузькому розумінні, проте може створити умови для порушення доступності, обходу фізичного контролю або подальшої атаки на внутрішні сервіси.

Важливо підкреслити, що рівень STS-моделі не є тотожним мережевому сегменту. Рівень визначає походження телеметрії та тип інфраструктурного контексту, тоді як сегмент може об’єднувати об’єкти з декількох рівнів моделі. Наприклад, окремий сегмент може містити сервери, мережеві правила, облікові записи, VPN-доступ, DNS-записи, фізичні пристрої та зовнішні перевірки доступності. Тому під час документування сегментів необхідно фіксувати не лише

IP-адреси або VLAN, а й пов’язані облікові записи, сервіси, джерела телеметрії, точки входу, дозволені потоки та можливі дії локалізації.

У межах STS-моделі доцільно розрізняти горизонтальне та вертикальне охоплення сегмента. Горизонтальне охоплення означає, що один рівень моделі включає багато однотипних сутностей, наприклад групу серверів, набір робочих станцій, VLAN користувачів або підмережу IP-камер. Вертикальне охоплення означає, що один бізнесовий або технічний сегмент проходить через декілька рівнів STS-моделі. Наприклад, сервіс віддаленого адміністрування може включати VPN-шлюз на рівні Border, внутрішні з’єднання на рівні Castle, сервери й облікові записи на рівні Moat, а також зовнішні перевірки доступності на рівні Shield Dome

Для практичного використання STS-моделі сегмент доцільно описувати за такими атрибутами: назва, призначення, критичність, пов’язані STS-рівні, активи, мережеві межі, дозволені та нетипові потоки, джерела телеметрії та можливі дії локалізації.

Таким чином, STS-модель дозволяє перейти від аналізу окремої події до аналізу її інфраструктурного контексту. Її основна цінність полягає у визначенні не лише місця виникнення сигналу, а й меж потенційної компрометації. Завдяки поділу джерел телеметрії на рівні, а також документуванню горизонтальних і вертикальних зв’язків між сегментами, модель створює основу для більш обґрунтованої локалізації інцидентів, пріоритезації перевірок і прийняття рішень щодо тимчасового обмеження або ізоляції окремих вузлів, сервісів чи каналів доступу

Core2.2 Особливості програмно-апаратної реалізації на рівні «Фортеця» та суміжних рівнях

У попередньому підрозділі було розглянуто основні рівні Structure-Based Telemetry Source Model (STS) та визначено їх роль у локалізації мережевих інцидентів. Однак для практичного застосування моделі важливо яким чином ці рівні можуть бути пов’язані з програмно-апаратною реалізацією системи

моніторингу. У межах даної роботи практична реалізація системи Castle Monitor найбільш тісно пов’язана з рівнем Castle / «фортеця», оскільки саме на цьому рівні аналізується внутрішня мережева взаємодія між вузлами локальної інфраструктури.

Рівень Castle / «фортеця» у STS-моделі не слід ототожнювати з окремим сервером, робочою станцією або мережевим сегментом. Його основне призначення полягає в описі взаємодії між внутрішніми сутностями інфраструктури: серверами, робочими станціями, сервісами, мережевими пристроями, обліковими записами та прикладними компонентами

У межах програмно-апаратної реалізації система Castle Monitor складається з декількох взаємопов’язаних компонентів: центрального серверного компонента Core API, бази даних PostgreSQL, агентів моніторингу, безагентних джерел телеметрії, інтерфейсу оператора та засобів візуалізації. Кожен із цих компонентів виконує окрему роль у зборі, передаванні, нормалізації та аналізі подій. Загальна ідея полягає в тому, що події з різних джерел надходять до центрального компонента, приводяться до єдиного формату, зберігаються у базі даних та використовуються для формування інцидентів. Приклад збору подій з різних рівнів зображено на рис. 2.1.

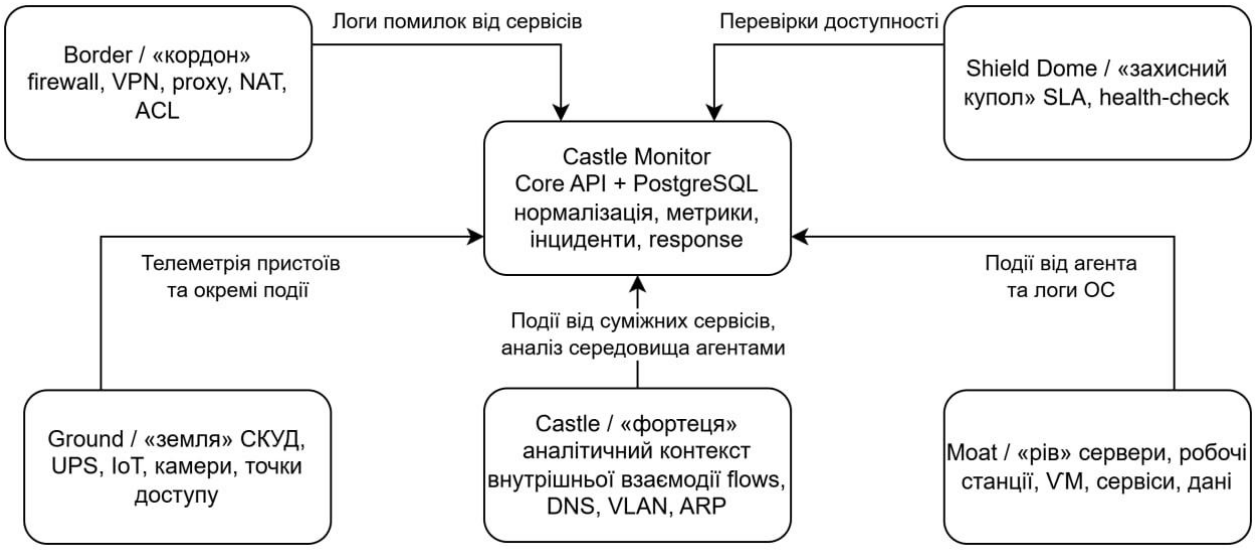


Рисунок 2.1 – Приклад збору подій
з девайсів на різних рівнях моделі STS.

Програмно-апаратна реалізація на рівні Castle / «фортеця» орієнтована передусім на виявлення поведінкових ознак, які складно визначити за однією ізольованою подією. Наприклад, одиничне підключення до сервера може бути легітимним, однак багаторазові підключення з одного вузла до великої кількості адрес або портів протягом короткого проміжку часу можуть свідчити про сканування. Аналогічно, окрема помилка автентифікації може бути випадковою, але серія невдалих спроб входу до одного або декількох сервісів може вказувати на атаку методом грубої сили. Саме тому система повинна аналізувати не тільки сам факт події, а й її повторюваність, напрямок, зв’язок з іншими подіями та місце в інфраструктурі.

Для опису практичної роботи рівня Castle / «фортеця» доцільно виділити декілька основних інформаційних потоків (табл. 2.2). Вони показують, як події з різних рівнів STS-моделі переходять у внутрішній аналітичний контекст системи.

Таблиця 2.2 – Основні інформаційні потоки системи на рівні Castle / «фортеця».

Інформаційний потік	**Джерело**	**Призначення**	**Приклад використання**
Локальні події вузла	Агент на сервері або робочій станції	Виявлення подій автентифікації, запуску служб, локальної активності та системних змін	Аналіз невдалих входів або запуску підозрілих процесів
Мережеві метадані	Агент, syslog, firewall або інше джерело	Виявлення напрямків з’єднань, повторюваних підключень і нетипових маршрутів	Виявлення сканування портів або горизонтального переміщення
Події міжмережевого екрана	Firewall, router, програмний маршрутизатор	Аналіз дозволених і заблокованих з’єднань, правил доступу та NAT	Виявлення repeated blocked access або порушення міжсегментної політики
Події віддаленого доступу	VPN-шлюз або proxy	Аналіз підключень користувачів і зовнішніх адрес	Виявлення підозрілого адміністративного входу

Інформаційний потік	Джерело	Призначення	Приклад використання
Метрики стану пристроїв	SNMP або інші джерела моніторингу	Контроль доступності обладнання та стану інтерфейсів	Виявлення відмови пристрою, втрати зв’язку або проблем із живленням
Нормалізовані події	Core API після обробки raw events	Приведення різнорідної телеметрії до єдиного формату	Подальше формування метрик, правил і інцидентів
Інциденти	Механізм виявлення системи	Узагальнення декількох пов’язаних подій у єдиний об’єкт реагування	Формування інциденту brute-force або scan activity
Дії реагування	Policy Guard та response-механізм	Підготовка безпечної реакції на інцидент	Створення planned або dry-run дії без негайного блокування критичних ресурсів

З погляду апаратного забезпечення рівень Castle / «фортеця» не потребує окремого спеціалізованого мережевого пристрою, оскільки його логіка реалізується через сукупність програмних компонентів і джерел телеметрії. Основне навантаження припадає на серверну частину, яка приймає події, зберігає їх, виконує нормалізацію та кореляцію. Агентська частина має бути достатньо легкою, щоб працювати на звичайних серверах, робочих станціях або малоресурсних вузлах. Детальні розрахункові вимоги до апаратного забезпечення серверної та клієнтської частини розглянуто в підрозділі 2.4.

Таким чином, особливість програмно-апаратної реалізації на рівні Castle / «фортеця» полягає у тому, що цей рівень не є окремим фізичним компонентом, а виступає центральним аналітичним контекстом системи. Його робота забезпечується агентами, безагентними джерелами телеметрії, Core API, базою даних та механізмами нормалізації подій. Завдяки цьому система Castle Monitor може поєднувати події з різних рівнів STS-моделі, виявляти поведінкові ознаки аномальної активності та формувати основу для подальшого реагування на інциденти.

2.3 Архітектура системи Castle Monitor

Після визначення рівнів моделі STS та ролі рівня Castle / «фортеця» доцільно перейти до опису загальної архітектури розробленої системи Castle Monitor. Якщо STS-модель задає логіку класифікації джерел телеметрії та локалізації інцидентів, то архітектура системи визначає, яким чином ці дані збираються, передаються, обробляються, зберігаються та використовуються для виявлення аномальної активності.

Система Castle Monitor побудована за централізованою серверно-агентською архітектурою з можливістю підключення безагентних джерел телеметрії. Центральним компонентом є серверна частина, яка приймає події, виконує їх нормалізацію, зберігає дані, формує інциденти та забезпечує взаємодію з інтерфейсом оператора. Агентська частина встановлюється на кінцеві вузли інфраструктури та відповідає за збір локальної телеметрії, передавання подій до центрального компонента, отримання політик і виконання дозволених дій реагування. Безагентні джерела використовуються для приймання телеметрії від мережевого обладнання, міжмережевих екранів, маршрутизаторів, систем виявлення вторгнень та інших пристроїв, на які недоцільно або неможливо встановити агент.

Загальну архітектуру системи Castle Monitor наведено на рис. 2.2.

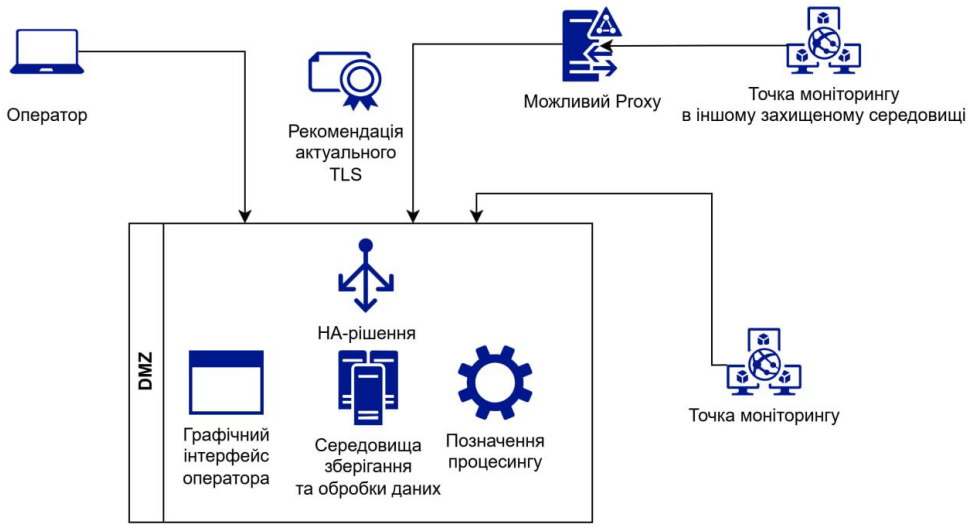


Рисунок 2.2 – Загальна архітектура системи Castle Monitor

До складу системи входять такі основні компоненти: центральний компонент Core API, база даних PostgreSQL, агент моніторингу, інтерфейс оператора, Grafana та джерела телеметрії. Кожен із компонентів виконує окрему функцію, однак практична цінність системи досягається саме завдяки їх взаємодії. Загальна логіка роботи системи полягає у тому, що подія виникає на кінцевому вузлі або мережевому пристрої, передається до Core API, зберігається у базі даних, нормалізується, аналізується правилами виявлення, після чого може бути сформований інцидент і підготовлена дія реагування.

Центральний компонент Ядро (англ. Core) є основою серверної частини системи. Application Programming Interface (API) використовується для приймання подій від агентів, обробки запитів інтерфейсу оператора, роботи з політиками, формування інцидентів і передавання агентам необхідних конфігурацій або директив. Core API виконує роль посередника між джерелами телеметрії, базою даних, механізмами аналізу та інтерфейсами керування. Саме через цей компонент відбувається основна логіка обробки подій.

У межах Core API доцільно виділити декілька логічних модулів. Першим є модуль приймання подій, який відповідає за отримання телеметрії від агентів або безагентних джерел. Другим є модуль нормалізації, який приводить різнорідні події до уніфікованого представлення. Третім є модуль поведінкового аналізу, який використовує накопичені події та метрики для виявлення шаблонної активності зловмисників. Четвертим є модуль політик, який визначає, чи може система планувати або виконувати певні дії реагування. Окремо можна виділити модуль аудиту, який фіксує важливі адміністративні дії та рішення системи.

Агент моніторингу є клієнтським компонентом системи. Він встановлюється на кінцевий вузол, наприклад сервер, робочу станцію або інший обчислювальний пристрій, і виконує локальний збір даних. Агент може передавати до Core API інформацію про стан вузла, системні журнали, події автентифікації, службові події та локальні мережеві з’єднання. Крім того, агент має підтримувати періодичне надсилання сигналу стану, що дозволяє центральному компоненту контролювати

доступність і актуальний статус вузла.

Безагентні джерела телеметрії доповнюють агентський підхід. До таких джерел можуть належати міжмережеві екрани, маршрутизатори, програмні маршрутизатори, системи виявлення вторгнень, syslog-джерела та пристрої, доступні через Simple Network Management Protocol (SNMP).

Основні компоненти, їх призначення та основні також зібрано та продемонстровано в табл. 2.3.

Таблиця 2.3 – Основні компоненти архітектури Castle Monitor

Компонент	**Призначення**	**Основні функції**
Ядро (Core API)	Центральний серверний компонент системи	Приймання подій, нормалізація, формування інцидентів, робота з агентами, політиками та діями реагування
База данних	Основне сховище даних	Збереження агентів, подій, метрик, інцидентів, політик, дій реагування та аудиту
Агент моніторингу	Клієнтський компонент на кінцевому вузлі	Збір локальної телеметрії, передавання подій, heartbeat, отримання політик і директив
Безагентні джерела	Джерела телеметрії без встановлення агента	Передавання syslog-подій, SNMP-метрик, firewall logs, VPN logs та іншої інфраструктурної телеметрії
Інтерфейс оператора	Засіб керування системою	Перегляд агентів, інцидентів, політик, дій реагування та службової інформації
Механізм політик реагування	Логіка безпечного планування дій	Перевірка дозволених і заборонених дій, dry-run, approval, protected ports, never block
Механізм виявлення	Аналітичний модуль системи	Виявлення шаблонної активності, побудова метрик, застосування правил і формування інцидентів

Загальний цикл обробки події у системі можна подати у вигляді послідовності: виникнення події на джерелі телеметрії, передавання події до Core API, збереження початкового запису, нормалізація, формування метрик, застосування правил виявлення, створення інциденту та підготовка дії реагування (рис. 2.3). Така послідовність дозволяє забезпечити простежуваність події від моменту її виникнення до моменту прийняття рішення системою.

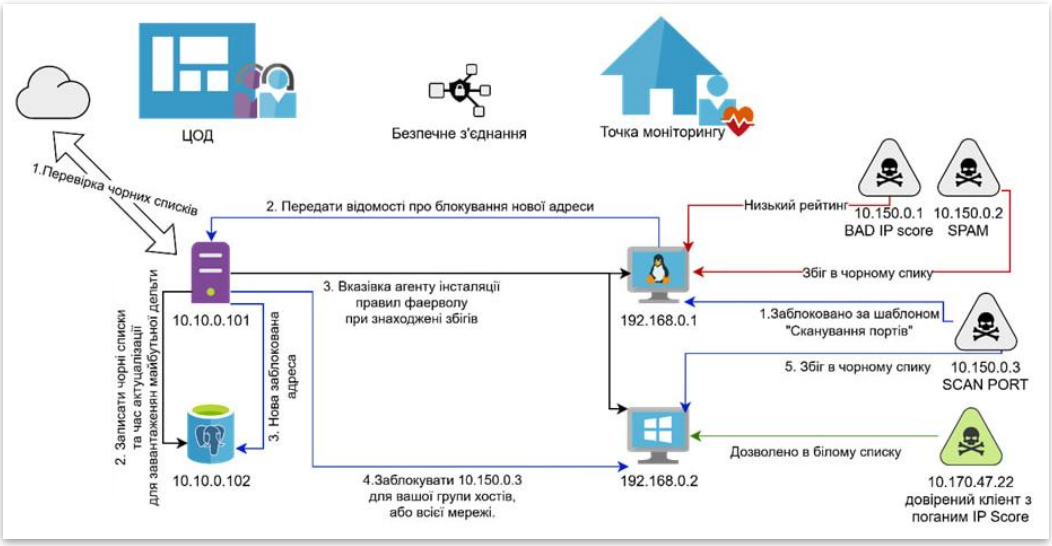


Рисунок 2.3 – Загальний цикл обробки події у системі Castle Monitor

У межах архітектури Castle Monitor для безпечного реагування використовуються такі принципи: режим симуляції дії за замовчуванням, потреба в підтвердженні для небезпечних дій, заборона блокування захищених адрес або портів, фіксація дій в аудиті та виконання локального реагування через агента. Це дозволяє розглядати систему як засіб керованого реагування на основі політик.

Архітектура системи також передбачає можливість масштабування. У базовому варіанті всі серверні компоненти можуть бути розгорнуті в межах одного тестового стенда. У більшому середовищі Core API, PostgreSQL, frontend можуть бути винесені на окремі сервери або віртуальні машини. Агентська частина може встановлюватися на різні типи вузлів, а безагентні джерела можуть передавати телеметрію централізовано. Це дозволяє поступово розширювати систему без повної зміни її архітектури.

Разом із цим архітектура має певні обмеження. Повнота аналізу залежить від якості телеметрії, правильності налаштування агентів, доступності журналів, актуальності інвентаризації та коректності правил виявлення. Крім того, частина джерел може передавати лише обмежений набір даних, що ускладнює побудову повної картини інциденту. Тому система повинна розглядатися як аналітичний і реагувальний шар, який підсилює наявні засоби моніторингу, але не скасовує потребу в якісній сегментації, інвентаризації активів і базових політиках безпеки.

2.4 Розрахункові вимоги до апаратного забезпечення сервера та клієнтських вузлів

Для практичного впровадження системи Castle Monitor важливо визначити не лише її програмну архітектуру, а й орієнтовні вимоги до апаратного забезпечення серверної та клієнтської частини. Такі вимоги необхідні для оцінювання можливості розгортання системи в локальній мережі, планування ресурсів, вибору обладнання та визначення меж масштабування. У цьому підрозділі наведено саме розрахункові вимоги до компонентів системи. Фактичні характеристики обладнання, використаного під час експериментальної перевірки, доцільно подати окремо в розділі 3.

Для невеликого лабораторного або навчального стенда достатньо одного серверного вузла з 4 ядрами CPU, 8 Гбайт RAM і SSD-накопичувачем від 100 Гбайт. Така конфігурація дозволяє розгорнути Core API, PostgreSQL, frontend в одному середовищі й перевірити основні функції системи: реєстрацію агентів, приймання подій, нормалізацію, формування інцидентів і відображення результатів. Для середовища з більшою кількістю агентів рекомендовано збільшити обсяг оперативної пам’яті до 16–32 Гбайт, а також передбачити окремий обсяг дискового простору для бази даних і журналів.

Орієнтовні вимоги до серверної та клієнтської частини наведено в табл. 2.4. Значення є розрахунковими та можуть уточнюватися залежно від фактичної інтенсивності подій, строку зберігання даних і кількості активних джерел телеметрії.

Таблиця 2.4 – Розрахункові вимоги до апаратного забезпечення Castle Monitor

Компонент	Мінімальні вимоги для лабораторного стенда	Рекомендовані вимоги для розширеного середовища	Призначення ресурсу
Сервер Castle Monitor	4 ядра CPU, 8 Гбайт RAM, SSD від 100 Гбайт, мережа 1 Гбіт/с	8 ядер CPU, 16–32 Гбайт RAM, SSD/NVMe від 250–500 Гбайт, мережа 1 Гбіт/с або вище	Робота Core API, PostgreSQL, frontend, Grafana, приймання та обробка подій

Компонент	Мінімальні вимоги для лабораторного стенда	Рекомендовані вимоги для розширеного середовища	Призначення ресурсу
База даних PostgreSQL	Може працювати на спільному сервері з Core API	Бажано винести на окремий сервер або віртуальну машину при зростанні кількості агентів	Збереження raw events, normalized events, метрик, інцидентів, політик і аудиту
Клієнтський вузол з агентом	1–2 ядра CPU, 512 Мбайт–1 Гбайт вільної RAM, 1-2 Гбайт дискового простору	2 ядра CPU, 2 Гбайт RAM, 5 Гбайт дискового простору для локальної черги та журналів	Збір локальної телеметрії, heartbeat, передавання подій, отримання політик
Малоресурсний клієнтський вузол	Одноплатний комп’ютер класу Raspberry Pi або подібний ARM-пристрій	ARM-пристрій з 2-4 Гбайт RAM і стабільним накопичувачем	Демонстрація можливості роботи агента в середовищах з обмеженими ресурсами
Операторський комп’ютер	Сучасний браузер, доступ до frontend і Grafana	Сучасний браузер, стабільне мережеве підключення	Перегляд інцидентів, агентів, політик, дашбордів і результатів реагування

Практична перевірка лабораторного розгортання показує, що в режимі простою серверна частина системи не створює значного навантаження на ресурси вузла. На рис. 2.4 наведено приклад роботи Docker-стенда Castle Monitor, у якому одночасно запущено контейнери Core API, PostgreSQL, syslog collector, frontend та додаткові розширення. У наведеному стані сумарне використання оперативної пам’яті контейнерами становить близько 831 Мбайт із доступних 7,42 Гбайт, а використання CPU залишається низьким. Це підтверджує, що для лабораторного стенда та початкової перевірки функціональності достатньо одного серверного вузла середньої продуктивності. Водночас ці значення не слід розглядати як остаточні вимоги для продуктивного середовища, оскільки під час активного надходження подій навантаження на Core API та PostgreSQL може суттєво зростати.

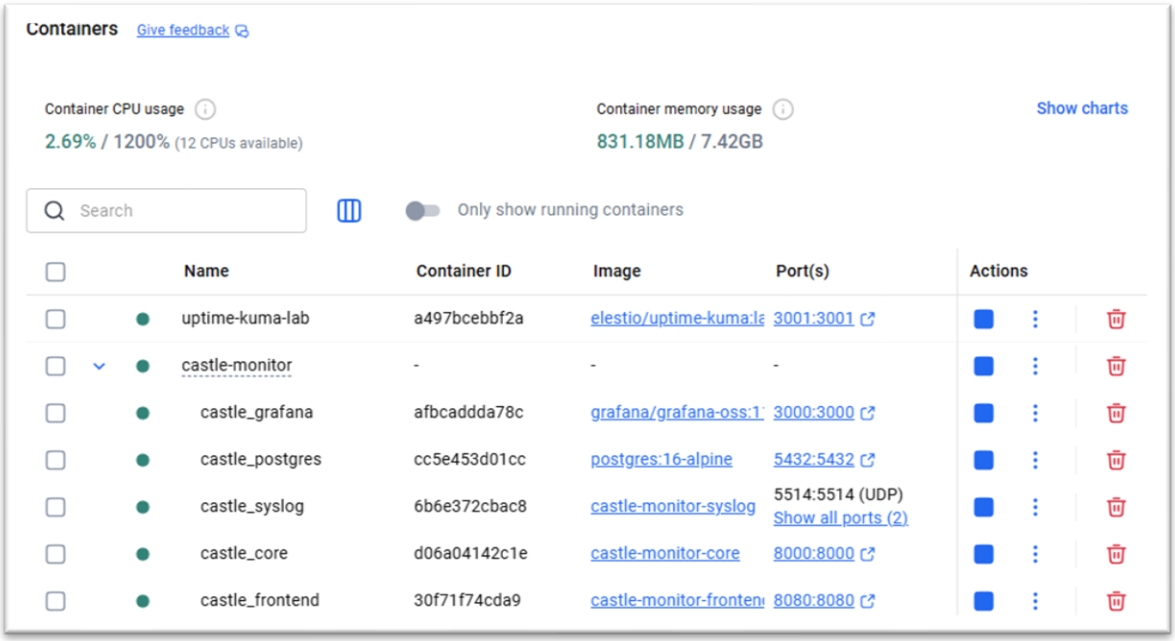


Рисунок 2.4 – Споживання ресурсів Docker-стенда Castle Monitor у режимі простою

Клієнтський агент не повинен вимагати значних ресурсів, оскільки його встановлення передбачається на робочі станції, сервери або інші вузли, які вже виконують власні прикладні задачі. Його основне навантаження пов’язане зі зчитуванням журналів, формуванням локальної черги подій, періодичним передаванням даних до Core API та отриманням політик. Тому агент має працювати у фоновому режимі та не повинен суттєво впливати на продуктивність основних сервісів вузла.

Окрему увагу слід приділити можливості використання агента на малоресурсних пристроях. У контексті даної роботи доцільно розглядати одноплатний комп’ютер класу Raspberry Pi як приклад клієнтського вузла з обмеженими ресурсами. Це не означає, що система повністю реалізує окрему платформу для інтернету речей (англ. Internet of Things, IoT), однак демонструє, що агентська архітектура може бути адаптована до пристроїв із невеликим обсягом оперативної пам’яті та обмеженою обчислювальною потужністю. Такий підхід є важливим для середовищ, де окрім серверів і робочих станцій присутні контролери, шлюзи, точки доступу або інші допоміжні пристрої.

Таким чином, апаратні вимоги до Castle Monitor визначаються масштабом локальної мережі, кількістю джерел телеметрії, інтенсивністю подій і строком зберігання даних. Для лабораторного стенда достатньо одного серверного вузла середньої продуктивності та декількох клієнтських вузлів з агентами. Для більшого середовища доцільно розділяти Core API, PostgreSQL, frontend та додаткові компоненти, а також передбачати запас дискового простору для зберігання подій. Агентська частина повинна залишатися легкою та придатною для роботи не лише на серверах і робочих станціях, а й на малоресурсних пристроях, що створює основу для подальшого використання системи в гетерогенних середовищах.

2.5 Організація збору, нормалізації та зберігання телеметрії

Ефективність системи моніторингу внутрішніх вразливостей локальної мережі значною мірою залежить від того, наскільки повно та узгоджено вона може збирати телеметрію з різних джерел. У межах системи Castle Monitor телеметрія розглядається як сукупність технічних даних, що описують стан вузлів, мережеві з’єднання, події автентифікації, службові повідомлення, події міжмережевих екранів, метрики доступності та інші сигнали, які можуть бути використані для виявлення аномальної активності.

Основною складністю під час роботи з такими даними є їх різнорідність. Події від операційних систем, міжмережевих екранів, систем виявлення вторгнень, мережевих пристроїв і агентів можуть мати різний формат, різний рівень деталізації та різну семантику. Наприклад, подія невдалої автентифікації в журналі операційної системи, запис про заблоковане з’єднання в міжмережевому екрані та повідомлення системи виявлення вторгнень можуть стосуватися одного інциденту, але бути представлені у різних форматах. Тому для подальшого аналізу такі події необхідно привести до єдиного логічного представлення.

Збір подій у системі організовано таким чином, щоб зберігати як початковий вигляд події, так і її нормалізоване представлення. Початкові події (англ. raw events) містять вихідний запис, отриманий від агента, syslog-джерела, мережевого

пристрою або іншого компонента. Збереження таких подій важливе для аудиту, повторної перевірки, уточнення правил обробки та можливості подальшого аналізу. Нормалізовані події (англ. normalized events) є результатом обробки початкових записів і містять уніфікований набір полів, придатних для кореляції, побудови метрик і застосування правил виявлення.

Нормалізація подій передбачає виділення ключових параметрів, які мають значення для подальшого аналізу. До таких параметрів належать час виникнення події, джерело телеметрії, тип події, рівень критичності, адреса джерела, адреса призначення, порт джерела, порт призначення, протокол, ідентифікатор агента, назва хоста, обліковий запис, назва сервісу та інші атрибути залежно від типу події. Завдяки цьому події з різних джерел можуть оброблятися за спільною логікою.

Окреме значення має контекст мережевого потоку, який у роботі розглядається через набір п’яти параметрів: адреса джерела, адреса призначення, порт джерела, порт призначення та протокол. Такий набір часто називають 5-tuple або flow-контекстом. У межах даної роботи ці поля використовуються як контекст події або інциденту, що допомагає описати напрямок взаємодії між вузлами.

Для наглядної процес було проілюстровано у вигляді схеми на рис. 2.5.

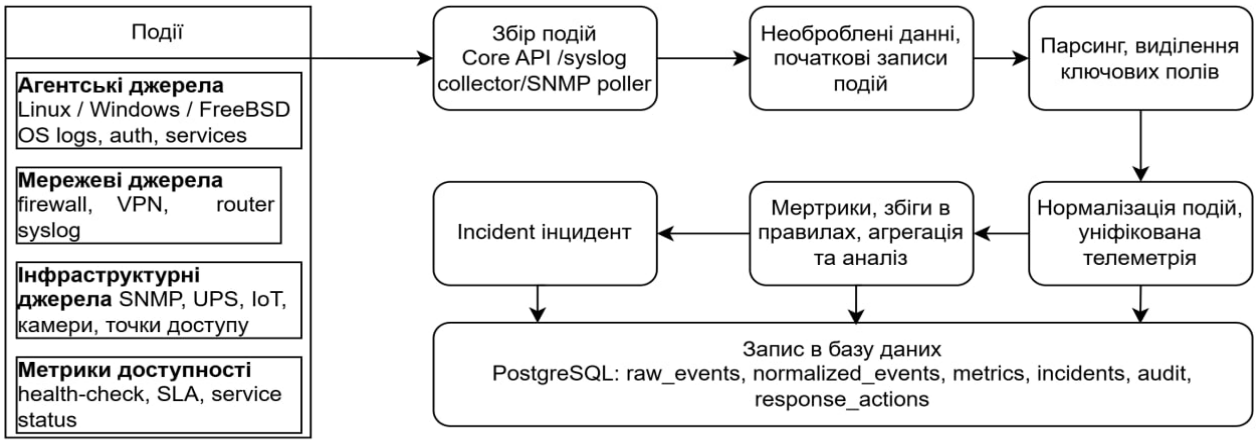


Рисунок 2.5 – Схема збору, нормалізації та зберігання телеметрії

Узагальнено процес обробки телеметрії можна описати як послідовність етапів: отримання події від джерела, збереження початкового запису, визначення типу події, виділення ключових полів, формування нормалізованої події, побудова

метрик і передавання результатів до механізму виявлення. Така послідовність дозволяє зберегти як вихідний контекст події, так і її уніфіковане представлення для подальшої аналітики.

Для прикладу, окремий невдалий вхід може бути звичайною помилкою користувача, однак велика кількість невдалих спроб за короткий проміжок часу може вказувати на атаку методом грубої сили (англ. brute-force). Для цього початкові записи журналів повинні бути приведені до такого вигляду, у якому система може порівнювати події між собою, обчислювати кількість спроб, визначати обліковий запис, джерело підключення та часовий інтервал.

Важливою вимогою є також обмеження строку зберігання телеметрії. Початкові події можуть займати значний обсяг дискового простору, особливо в умовах активного збору журналів або під час інцидентів, коли кількість подій різко зростає. Тому для практичного використання системи необхідно передбачати політики зберігання, очищення або архівування старих записів.

Таким чином, організація збору, нормалізації та зберігання телеметрії є основою для подальшого поведінкового аналізу. Без приведення подій до єдиного формату система не зможе коректно виявляти повторювані шаблони активності, зіставляти події з різних джерел і формувати обґрунтовані інциденти. Саме тому в архітектурі Castle Monitor процес нормалізації розглядається як проміжний, але критично важливий етап між отриманням події та прийняттям рішення щодо реагування.

2.6 Механізм поведінкового аналізу та реагування на інциденти

Після збору та нормалізації телеметрії наступним етапом роботи системи є поведінковий аналіз подій. На відміну від підходу, за якого кожна подія розглядається ізольовано, поведінковий аналіз орієнтований на виявлення повторюваних або нетипових шаблонів активності. Саме такі шаблони часто є більш показовими для виявлення внутрішніх мережевих інцидентів, ніж окремі записи журналів.

У межах системи Castle Monitor поведінковий аналіз використовується для виявлення шаблонної активності зловмисників. До таких шаблонів можуть належати повторні заблоковані підключення, горизонтальне сканування, вертикальне сканування портів, атаки методом грубої сили, нетипове збільшення кількості з’єднань, підозріла активність між сегментами та аномальна взаємодія з внутрішніми сервісами. Кожен із цих сценаріїв може бути слабко виражений на рівні окремої події, але стає помітним після накопичення й кореляції декількох подій за певний проміжок часу.

Наприклад, горизонтальне сканування пов’язане зі спробами одного джерела звертатися до багатьох вузлів у межах мережі або сегмента. Така поведінка може свідчити про пошук доступних хостів або сервісів після первинної компрометації. Вертикальне сканування, навпаки, полягає у зверненні до багатьох портів одного вузла, що може вказувати на спробу визначити перелік відкритих сервісів. Обидва сценарії є важливими для рівня Castle / «фортеця», оскільки вони описують нетипову внутрішню мережеву взаємодію.

Окремим напрямом може бути використання індикаторів компрометації (англ. Indicators of Compromise, IOC). До них можуть належати підозрілі IP-адреси, доменні імена, URL-адреси або інші ознаки, пов’язані з відомими загрозами. У межах поточної логіки IOC доцільно розглядати як додатковий фактор ризику, а не як єдину підставу для автоматичного блокування. Це особливо важливо, оскільки помилкове спрацювання на основі зовнішнього індикатора може призвести до небажаного обмеження легітимної взаємодії.

Результатом поведінкового аналізу є формування інциденту (рис. 2.6). Інцидент у системі Castle Monitor не повинен розглядатися як копія однієї події. Він є узагальненням однієї або декількох пов’язаних подій, які разом утворюють підозрілий сценарій. Інцидент має містити тип виявленої активності, джерело, ціль, час виникнення, рівень критичності, ступінь упевненості, пов’язані події та контекст, необхідний для подальшої перевірки оператором або механізмом реагування.

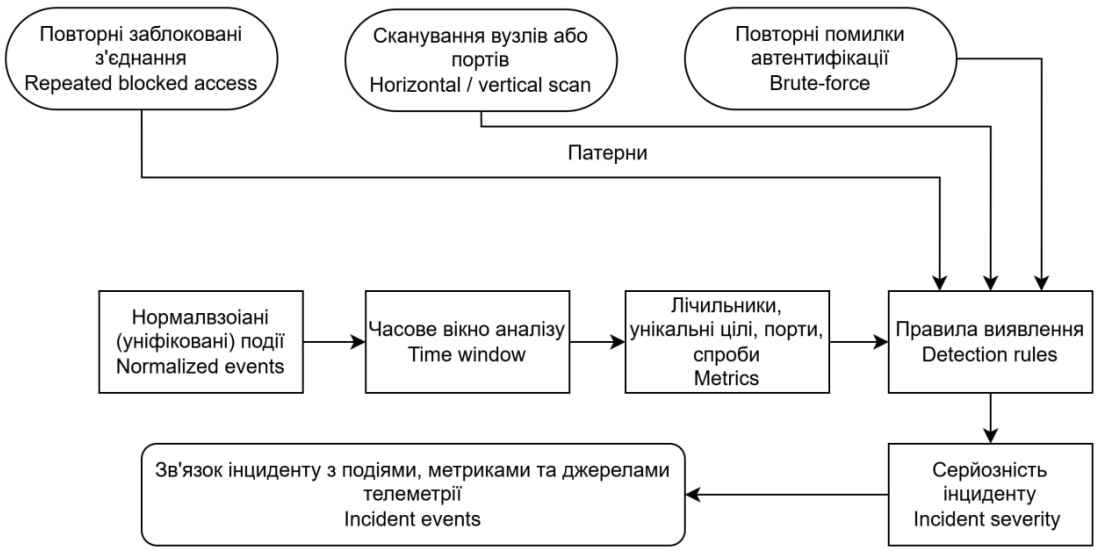


Рисунок 2.6 – Послідовність формування інциденту на основі поведінкового аналізу

Узагальнено процес формування інциденту можна описати так: нормалізовані події надходять до механізму аналізу, на їх основі формуються метрики, правила виявлення перевіряють наявність підозрілих шаблонів, після чого створюється інцидент із посиланням на відповідні події. Такий підхід дозволяє уникнути ситуації, коли система реагує на кожну окрему подію без урахування контексту.

Після формування інциденту система може перейти до етапу реагування. У межах Castle Monitor реагування має бути безпечним за замовчуванням. Це означає, що система не повинна одразу виконувати небезпечні дії, які можуть порушити роботу критичних сервісів, адміністративного доступу або службових вузлів. Перед виконанням дії необхідно перевірити політики реагування, режим роботи системи, наявність захищених адрес, захищених портів і потребу в підтвердженні оператором.

Механізм політик реагування виконує роль захисного шару між виявленням інциденту та виконанням дії. Він визначає, які дії дозволені, які заборонені, чи потрібне підтвердження оператора, чи має дія виконуватися в режимі симуляції, який строк дії правила та які адреси або порти не можна блокувати. Такий підхід дозволяє зменшити ризик помилкового реагування.

Окрему роль відіграють винятки, які не можна блокувати автоматично. До них можуть належати джерела адміністрування, критичні сервери, службові адреси, захищені порти та вузли, що забезпечують доступ до самої системи моніторингу. Це дозволяє запобігти ситуації, коли система помилково блокує адміністративний доступ або критичний інфраструктурний сервіс.

Таким чином, механізм поведінкового аналізу та реагування в системі Castle Monitor поєднує декілька послідовних етапів: накопичення нормалізованих подій, формування метрик, виявлення шаблонної активності, створення інциденту, перевірку політик реагування та підготовку контрольованої дії. Завдяки цьому система не обмежується фіксацією окремих подій, а дозволяє аналізувати поведінку в часі та приймати більш обґрунтовані рішення щодо локалізації загрози.

Висновки до розділу 2

У другому розділі було розроблено проєктні рішення для системи моніторингу внутрішніх вразливостей локальної мережі Castle Monitor. Основою запропонованого підходу стала Structure-Based Telemetry Source Model (STS), яка використовується для структуризації джерел телеметрії, визначення рівня походження подій та локалізації потенційної зони компрометації. Модель дозволяє розглядати інцидент не як ізольовану подію, а як сигнал, пов’язаний із певним рівнем інфраструктури, суміжними вузлами, сервісами та мережевими сегментами.

Було визначено роль рівня Castle / «фортеця» як центральної аналітичної площини для аналізу внутрішньої мережевої взаємодії. Показано, що для виявлення сканування, атак методом грубої сили, бокового переміщення зловмисника та аномального east-west трафіку необхідно поєднувати події з агентських і безагентних джерел телеметрії, зокрема системних журналів, міжмережевих екранів, syslog-повідомлень, SNMP-метрик та мережевих подій.

У межах розділу було спроєктовано архітектуру Castle Monitor, яка включає Core API, базу даних, агентів моніторингу, безагентні джерела телеметрії. Описано загальний цикл обробки події: від збору та нормалізації телеметрії до формування метрик, виявлення інциденту та підготовки дії реагування.

3 РЕАЛІЗАЦІЯ ТА ПЕРЕВІРКА СИСТЕМИ CASTLE MONITOR

3.1 Реалізація основних компонентів системи

Після визначення архітектури, рівнів Structure-Based Telemetry Source Model (STS), вимог до апаратного забезпечення та логіки обробки телеметрії було виконано практичну реалізацію основних компонентів системи Castle Monitor. Метою практичної реалізації було створення працездатного прототипу системи, здатної приймати події від агентів і безагентних джерел, зберігати телеметрію, виконувати її нормалізацію, формувати інциденти та підтримувати безпечне реагування на виявлену активність.

Реалізація системи виконана за модульним принципом. Основні функції розділено між центральним серверним компонентом, базою даних, агентською частиною, інтерфейсом оператора та засобами візуалізації. Такий підхід спрощує розгортання системи, її супровід і подальше розширення. Крім того, модульна структура дозволяє змінювати або доповнювати окремі компоненти без повної перебудови всієї системи.

Основним серверним компонентом є Core API. Він виконує роль центральної точки обміну між агентами, базою даних, інтерфейсом оператора та механізмами аналізу (рис. 3.1). Через Core API відбувається реєстрація агентів, приймання подій, обробка heartbeat-повідомлень, надання політик агентам, збереження телеметрії та формування службових відповідей. У межах реалізації Core API також відповідає за роботу з інцидентами, політиками реагування та діями реагування.

Для зберігання даних використано PostgreSQL. База даних містить службові та аналітичні сутності системи: агентів, реєстраційні токени, події, метрики, інциденти, правила виявлення, політики реагування, дії реагування та записи аудиту. Використання PostgreSQL дозволяє зберігати зв’язки між подіями, джерелами телеметрії та інцидентами, що є важливим для подальшого аналізу й пояснення логіки роботи системи. Фактична структура бази даних детальніше розглядається в підрозділі 3.2.

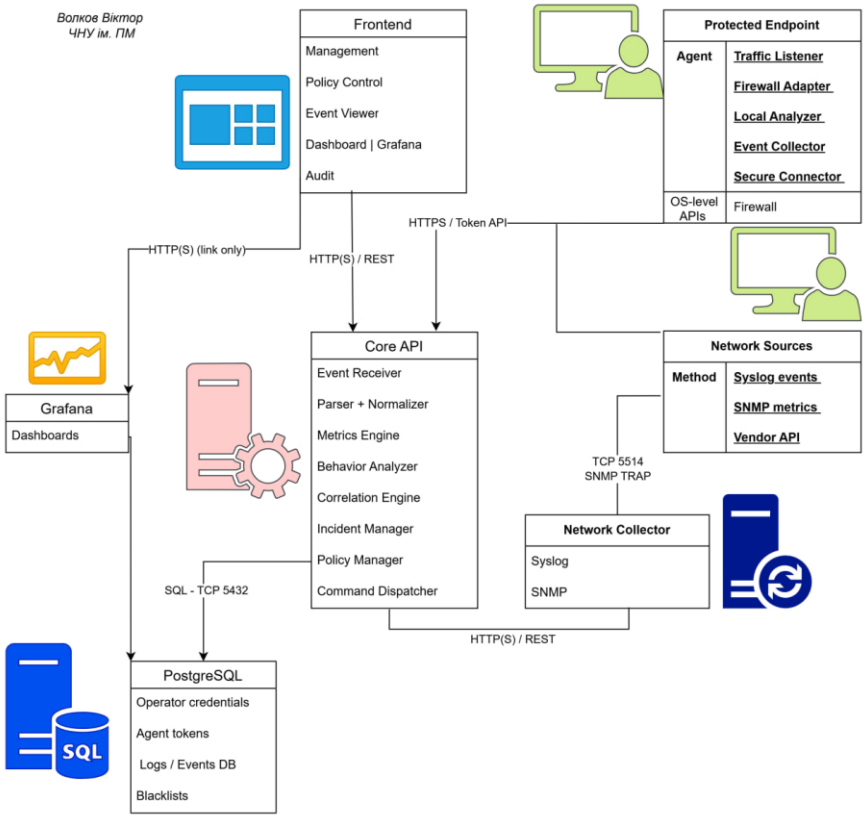


Рисунок 3.1 – Загальна структура реалізованих компонентів Castle Monitor

Окремим компонентом є агент моніторингу. Агент призначений для встановлення на кінцевий вузол локальної мережі. Його основні функції полягають у зборі локальної телеметрії, передаванні подій до Core API, надсиланні heartbeat-повідомлень, отриманні політик та виконанні дозволених дій реагування. Агентська модель важлива для системи, оскільки саме вона дозволяє отримувати події з нижніх рівнів STS-моделі, тобто з серверів, робочих станцій, віртуальних машин, контейнерів і сервісів.

У межах реалізації агент підтримує життєвий цикл підключення до системи (рис. 3.2). Перед початком повноцінної роботи агент має пройти етап початкової реєстрації, після чого адміністратор або оператор може підтвердити його використання в системі. Такий підхід дозволяє уникнути неконтрольованого підключення сторонніх вузлів і забезпечує базову керованість агентської інфраструктури. Після підтвердження агент може передавати події, повідомляти про стан вузла та отримувати необхідні конфігурації.

Рисунок 3.2 – Життєвий цикл агента моніторингу в системі Castle Monitor

Для роботи з мережевими та інфраструктурними джерелами передбачено безагентний підхід. Він використовується для пристроїв і систем, на які неможливо або недоцільно встановлювати агент. До таких джерел належать міжмережеві екрани, маршрутизатори, syslog-джерела, Simple Network Management Protocol (SNMP)-пристрої, Virtual Private Network (VPN)-шлюзи та інші елементи інфраструктури. Безагентні джерела доповнюють агентську телеметрію та дозволяють отримувати дані з рівнів Border / «кордон» і Ground / «земля».

Інтерфейс оператора реалізовано як окремий frontend-компонент. Його призначення полягає у наданні зручного доступу до основних сутностей системи: агентів, інцидентів, політик, дій реагування та службових станів. Інтерфейс оператора не виконує самостійний аналіз подій, а працює з даними через Core API. Такий підхід дозволяє розділити логіку обробки подій і логіку відображення інформації користувачу.

Додатково, для демонстрації, використовується Grafana, яка виконує роль засобу візуалізації. Вона призначена для відображення агрегованих показників, динаміки подій, стану компонентів і статистики інцидентів. Grafana не замінює інтерфейс оператора, оскільки має інше призначення: вона зручна для перегляду

графіків, часових рядів і дашбордів (англ. dashboards), тоді як frontend використовується для операційної роботи із сутностями системи.

У лабораторному середовищі основні серверні компоненти системи розгорнуто у контейнерах (рис. 3.3). Контейнеризація спрощує запуск системи, дозволяє ізолювати компоненти один від одного та забезпечує відтворюваність середовища. До складу Docker-стенда входять контейнери Core API, PostgreSQL, frontend, Grafana та syslog-компонента. Така структура дозволяє швидко підняти стенд для перевірки функціональності та продемонструвати взаємодію між компонентами.

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\Viktor> docker ps
CONTAINER ID   IMAGE                          COMMAND                  CREATED         STATUS                   PORTS                                                                                  NAMES
30f71f74cda9   castle-monitor-frontend        "/docker-entrypoint.…"   3 hours ago     Up 3 hours               0.0.0.0:8080->8080/tcp, [::]:8080->8080/tcp                                            castle_frontend
d06a04142c1e   castle-monitor-core            "sh /docker-entrypoi…"   3 hours ago     Up 3 hours (healthy)     0.0.0.0:8000->8000/tcp, [::]:8000->8000/tcp                                            castle_core
a497bcebbf2a   elestio/uptime-kuma:latest     "/usr/bin/dumb-init …"   4 hours ago     Up 4 hours (healthy)     0.0.0.0:3001->3001/tcp, [::]:3001->3001/tcp                                            uptime-kuma-lab
6b6e372cbac8   castle-monitor-syslog          "python main.py"         5 hours ago     Up 5 hours               0.0.0.0:5514->5514/udp, [::]:5514->5514/udp, 0.0.0.0:5515->5515/tcp, [::]:5515->5515/tcp   castle_syslog
afbcaddda78c   grafana/grafana-oss:11.5.2     "/run.sh"                8 days ago      Up 2 days                0.0.0.0:3000->3000/tcp, [::]:3000->3000/tcp                                            castle_grafana
cc5e453d01cc   postgres:16-alpine             "docker-entrypoint.s…"   8 days ago      Up 2 days (healthy)      0.0.0.0:5432->5432/tcp, [::]:5432->5432/tcp                                            castle_postgres
```

Рисунок 3.3 – Стан контейнерів Castle Monitor у тестовому Docker-середовищі

У режимі простою лабораторний стенд демонструє помірне споживання ресурсів. Це важливо для підтвердження того, що система може бути розгорнута в навчальному або тестовому середовищі без використання потужного серверного обладнання. Водночас показники простою не є остаточною характеристикою продуктивності системи, оскільки під час активного надходження подій, навантаження може суттєво зростати.

Практична реалізація системи також передбачає розділення етапів виявлення інциденту та реагування на нього. Це зроблено для зменшення ризику помилкового блокування критичних сервісів або адміністративного доступу.

Механізм реагування реалізовано з урахуванням принципу безпечної поведінки за замовчуванням. Для цього використовуються режим симуляції дії (англ. dry-run), підтвердження оператором для небезпечних дій, обмеження для

захищених портів, джерел адміністрування і адрес, які не можна блокувати автоматично. Такий підхід дозволяє використовувати систему не лише для виявлення інцидентів, а й для контрольованої підготовки дій локалізації.

У межах реалізації доцільно виділити такі основні результати: створено центральний компонент Core API, підготовлено структуру бази даних PostgreSQL, реалізовано агентську взаємодію, передбачено безагентне отримання телеметрії, створено інтерфейс оператора, інтегровано Grafana для візуалізації, реалізовано збереження подій та основу для формування інцидентів і дій реагування.

Таким чином, у межах практичної реалізації було створено працездатну основу системи Castle Monitor, яка поєднує серверну частину, базу даних, агентський компонент, безагентні джерела телеметрії, інтерфейс оператора та засоби візуалізації. Така реалізація відповідає проєктним рішенням, описаним у розділі 2, і створює основу для подальшої перевірки працездатності системи в лабораторному середовищі.

3.2 Реалізація бази даних та інформаційної моделі

Одним із ключових компонентів системи Castle Monitor є база даних, яка забезпечує збереження службової інформації, телеметрії, інцидентів, політик реагування та результатів виконання дій. Якщо Core API виконує роль центрального програмного компонента для приймання й обробки подій, то база даних є основою для збереження стану системи та забезпечення простежуваності подій від моменту їх отримання до формування інциденту або дії реагування.

Для реалізації сховища даних у системі використано PostgreSQL. Вибір реляційної бази даних обґрунтований необхідністю зберігати взаємопов’язані сутності: агентів, події, метрики, інциденти, політики, дії реагування та записи аудиту. Такий підхід дозволяє не лише накопичувати телеметрію, а й будувати зв’язки між джерелами подій, нормалізованими записами, сформованими інцидентами та рішеннями системи (рис. 3.4).

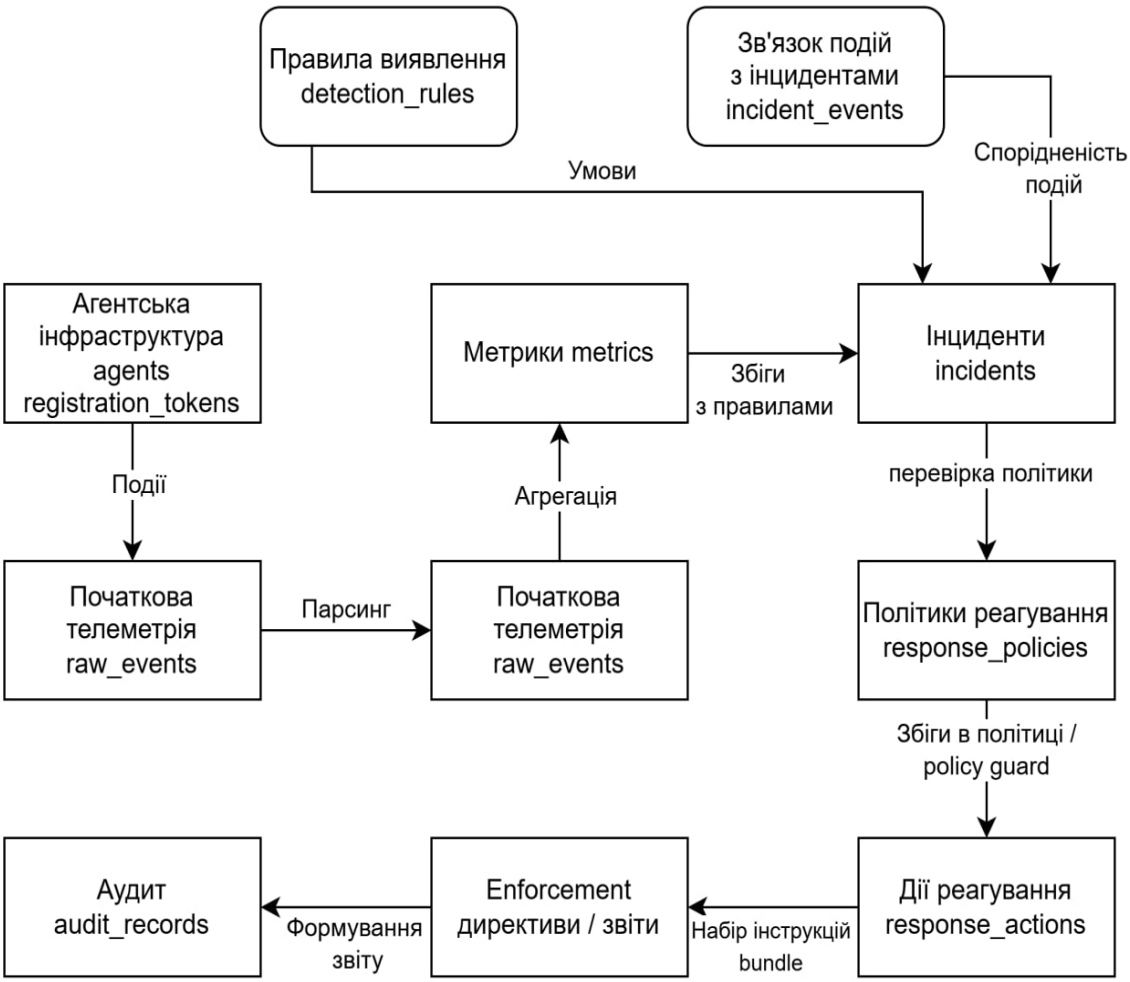


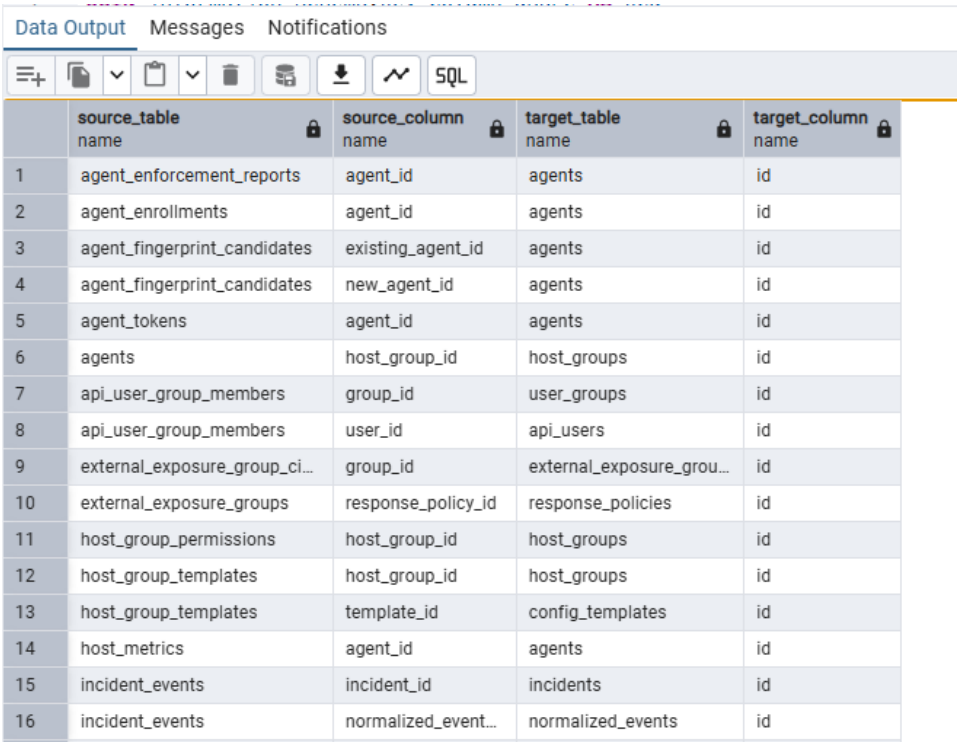
Рисунок 3.4 – Узагальнена інформаційна модель бази даних Castle Monitor

Інформаційна модель Castle Monitor побудована навколо декількох основних груп сутностей. Перша група відповідає за агентську інфраструктуру: агенти, реєстраційні токени, стан агента, heartbeat-повідомлення та службові параметри. Друга група пов’язана з телеметрією: початкові події, нормалізовані події та метрики. Третя група описує результати аналізу: правила виявлення, інциденти та зв’язки інцидентів із подіями. Четверта група відповідає за безпечне реагування: політики реагування, заплановані дії, enforcement-директиви та звіти про виконання. Окрему роль виконує аудит, який фіксує важливі адміністративні дії та рішення системи.

Узагальнене призначення основних сутностей бази даних наведено в табл. 3.1. Приклад такої таблиці наведено на рис. 3.5.

Таблиця 3.1 – Основні сутності бази даних Castle Monitor

Група сутностей	Приклади сутностей	Призначення
Агентська інфраструктура	agents, registration_tokens	Реєстрація агентів, ідентифікація вузлів, контроль стану та доступу
Аудит	audit_records	Фіксація адміністративних дій, службових змін і важливих рішень системи
Початкова телеметрія	raw_events	Збереження вихідних повідомлень від агентів, syslog, firewall, SNMP та інших джерел
Нормалізована телеметрія	normalized_events	Уніфіковане представлення подій для кореляції, метрик і правил виявлення
Метрики	metrics	Агреговані показники активності за певний проміжок часу
Виявлення	detection_rules	Опис правил або шаблонів, за якими система визначає підозрілу активність
Інциденти	incidents, incident_events	Збереження сформованих інцидентів і зв'язку між інцидентами та подіями
Політики реагування	response_policies	Перевірка дозволених і заборонених дій, dry-run, approval, protected ports, never block
Дії реагування	response_actions	Збереження запланованих, симульованих, підтверджених або виконаних дій
Enforcement-механізм	enforcement directives, enforcement reports	Передавання дозволених директив агенту та отримання результатів локального застосування


Data Output Messages Notifications

SQL

	source_table name	source_column name	target_table name	target_column name
1	agent_enforcement_reports	agent_id	agents	id
2	agent_enrollments	agent_id	agents	id
3	agent_fingerprint_candidates	existing_agent_id	agents	id
4	agent_fingerprint_candidates	new_agent_id	agents	id
5	agent_tokens	agent_id	agents	id
6	agents	host_group_id	host_groups	id
7	api_user_group_members	group_id	user_groups	id
8	api_user_group_members	user_id	api_users	id
9	external_exposure_group_ci...	group_id	external_exposure_grou...	id
10	external_exposure_groups	response_policy_id	response_policies	id
11	host_group_permissions	host_group_id	host_groups	id
12	host_group_templates	host_group_id	host_groups	id
13	host_group_templates	template_id	config_templates	id
14	host_metrics	agent_id	agents	id
15	incident_events	incident_id	incidents	id
16	incident_events	normalized_event...	normalized_events	id

Рисунок 3.5 – Приклад перегляду таблиці агентів у PostgreSQL / pgAdmin

Оскільки повна структура бази даних може містити значну кількість полів, індексів, обмежень і службових параметрів, її недоцільно повністю переносити в основний текст розділу. У межах підрозділу описано логіку зв’язків між сутностями та показано основні таблиці.

База даних забезпечує повний ланцюг простежуваності: джерело телеметрії → початкова подія → нормалізована подія → метрика → інцидент → дія реагування → аудит. Така послідовність є важливою для пояснюваності роботи системи. Оператор або дослідник може не лише побачити факт інциденту, а й перевірити, які саме події стали його основою та яку дію система запропонувала у відповідь.

Окреме значення має збереження аудиту. Аудит дозволяє фіксувати дії користувачів і системні зміни: створення або зміну політик, підтвердження агентів, планування дій реагування, зміну статусу інциденту або інші важливі операції. Для системи, яка може впливати на мережеве реагування, аудит є обов’язковим елементом, оскільки без нього складно визначити, хто і коли дозволив певну дію.

Під час проєктування бази даних також важливо враховувати майбутнє зростання обсягу телеметрії. Початкові та нормалізовані події можуть накопичуватися швидко, особливо в умовах активного сканування, атаки методом грубої сили або великої кількості агентів. Тому для практичного використання системи необхідні індекси за часом події, агентом, IP-адресами, типом події та ідентифікаторами інцидентів. Також важливими є механізми очищення або архівування старих подій.

3.3 Реалізація збору подій та роботи агентів

Агент моніторингу в системі Castle Monitor є окремим клієнтським компонентом, який встановлюється на кінцевий вузол локальної мережі. До таких вузлів можуть належати сервери, робочі станції, віртуальні машини або інші пристрої, здатні виконувати агентський процес. Основними завданнями агента є збір локальних подій, передавання телеметрії до центрального компонента Core

API, періодичне повідомлення про стан вузла, отримання політик і виконання дозволених дій реагування.

Життєвий цикл агента починається з етапу початкового підключення до системи. Для цього агент має отримати конфігурацію, адресу Core API та реєстраційні параметри. Після запуску агент надсилає запит на реєстрацію, унаслідок чого центральний компонент отримує інформацію про новий вузол. Надалі адміністратор або оператор може підтвердити агента, після чого він переходить у робочий стан і отримує можливість передавати події та отримувати політики. Процедуру замовлення також можна виконати через графічний інтерфейс (рис. 3.6).

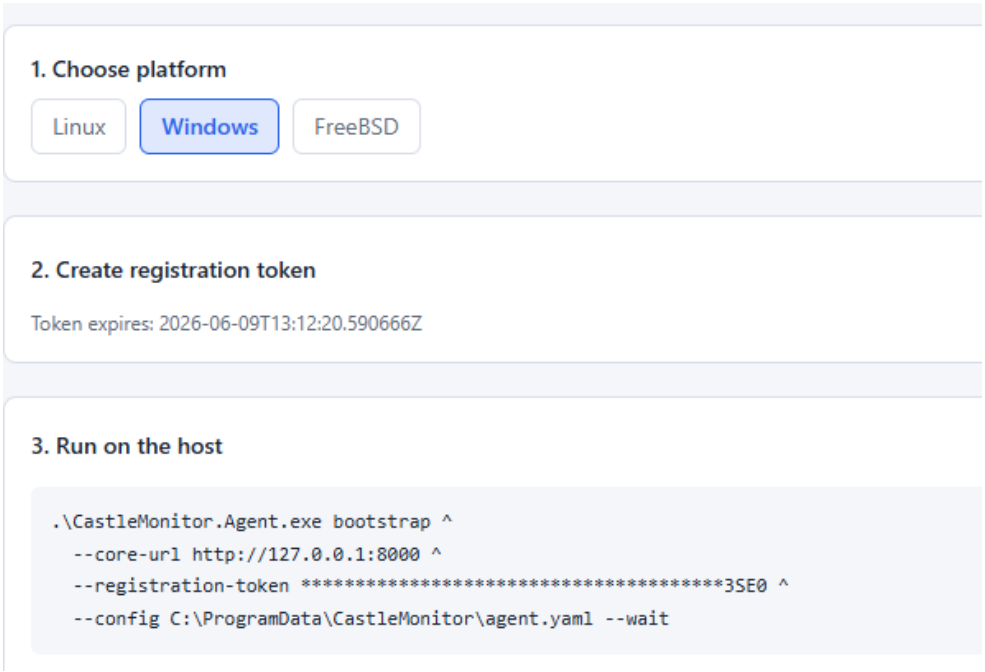


Рисунок 3.6 – Реєстрації агента через графічний інтерфейс

Процедура підтвердження агента є важливою з точки зору безпеки. Вона запобігає ситуації, коли сторонній або некоректно налаштований вузол починає передавати події до системи без контролю адміністратора. Після підтвердження агент ідентифікується у базі даних, має власний статус, ім’я, адресу, версію, тип операційної системи, режим роботи та параметри локального реагування. Це дозволяє пов’язувати події не лише з IP-адресою, а й з конкретним зареєстрованим вузлом.

Після реєстрації агент періодично надсилає heartbeat-повідомлення. Heartbeat використовується для контролю доступності агента, оновлення часу останньої активності та загального стану вузла. Завдяки цьому Core API може визначати, які агенти активні, які давно не надсилали дані, а які потребують перевірки (рис. 3.7). У практичному використанні це важливо, оскільки втрата зв’язку з агентом може бути як технічною проблемою, так і непрямою ознакою інциденту.

Рисунок 3.7 – Приклад статусу агента в інтерфейсі оператора

Основним джерелом телеметрії для агента є локальні журнали операційної системи. Для вузлів на базі Linux це можуть бути журнали автентифікації, системні журнали, службові повідомлення та інші файли журналів, доступні для читання агентом. Такі події можуть містити інформацію про невдалі спроби входу, запуск служб, помилки, зміну стану сервісів або іншу активність, яка має значення для виявлення інцидентів. Для коректної роботи агент повинен мати права на читання відповідних журналів.

Безагентні джерела телеметрії доповнюють роботу агентів. Вони використовуються для отримання даних від пристроїв і систем, на які не встановлюється агент. До таких джерел можуть належати міжмережеві екрани, маршрутизатори, програмні маршрутизатори, VPN-шлюзи, системи виявлення вторгнень, syslog-сервери та SNMP-пристрої. Події від таких джерел дозволяють аналізувати рівень Border / «кордон», а також частково рівень Ground / «земля», якщо йдеться про стан фізичного або периферійного обладнання.

Syslog-події можуть містити записи про заблоковані підключення, дозволені з’єднання, зміни стану інтерфейсів, спрацювання правил або службові повідомлення пристроїв. SNMP-метрики можуть використовуватися для отримання інформації про доступність пристрою, стан інтерфейсів, час роботи, лічильники трафіку або інші параметри. У поєднанні з агентськими подіями ці джерела дозволяють сформувати більш повну картину активності в локальній мережі.

Важливо, що агентський і безагентний підходи не замінюють один одного, а доповнюють. Агент дозволяє отримати детальні події з конкретного вузла, тоді як syslog, SNMP та події мережевого обладнання дозволяють бачити загальний контекст мережевої взаємодії. У поєднанні ці джерела створюють основу для поведінкового аналізу: система може зіставляти події автентифікації, мережеві підключення, заблоковані спроби доступу та стан інфраструктурних пристроїв.

3.4 Реалізація правил виявлення та формування інцидентів

Після організації збору подій від агентів і безагентних джерел наступним етапом роботи системи Castle Monitor є аналіз отриманої телеметрії та формування інцидентів. У межах реалізації система не обмежується збереженням окремих подій, а використовує їх для виявлення повторюваних шаблонів активності, які можуть свідчити про спробу сканування, атаку методом грубої сили, повторні заблоковані підключення або інші ознаки потенційної компрометації.

Основою для роботи механізму виявлення є нормалізовані події. На відміну від початкових записів, які можуть мати різний формат залежно від джерела, нормалізовані події містять уніфікований набір полів. До таких полів належать час події, джерело телеметрії, тип події, адреса джерела, адреса призначення, порт джерела, порт призначення, протокол, рівень критичності, ідентифікатор агента або пристрою, а також додаткові атрибути, які залежать від конкретного типу події. Саме така структура дозволяє застосовувати однакову логіку аналізу до подій, отриманих від різних джерел.

Механізм виявлення побудовано навколо правил, які аналізують не лише факт виникнення окремої події, а й її повторюваність, часовий контекст і зв’язок з іншими подіями. Такий підхід є важливим, оскільки одинична подія не завжди є ознакою інциденту.

У системі Castle Monitor правила виявлення можуть використовувати агреговані показники або метрики. Метрики формуються на основі нормалізованих подій і дозволяють оцінювати поведінку джерела або цілі за певне часове вікно. Наприклад, система може підраховувати кількість невдалих спроб автентифікації з однієї адреси, кількість унікальних вузлів, до яких звертається джерело, кількість портів, які перевіряються на одному вузлі, або кількість повторних заблокованих з’єднань. Завдяки цьому правила можуть працювати не з одиничними записами, а з поведінковими ознаками.

Одним із базових сценаріїв виявлення є repeated blocked access, тобто повторні заблоковані підключення. Такий сценарій може виникати, коли один вузол багаторазово намагається звернутися до забороненого ресурсу, закритого порту або іншого сегмента мережі. Окрема заблокована спроба не завжди є критичною, однак систематичне повторення таких подій може вказувати на сканування, помилкову конфігурацію або спробу обходу політик доступу.

Іншим важливим сценарієм є горизонтальне сканування. Воно проявляється у зверненнях одного джерела до багатьох різних вузлів мережі. Така поведінка характерна для пошуку активних хостів або сервісів після первинного проникнення в інфраструктуру. У межах Structure-Based Telemetry Source Model (STS) цей сценарій пов’язаний передусім з рівнем Castle / «фортеця», оскільки він описує нетипову внутрішню мережеву взаємодію між вузлами.

Вертикальне сканування відрізняється тим, що джерело звертається до багатьох портів одного цільового вузла. Такий сценарій може свідчити про спробу визначити відкриті сервіси або знайти вразливу службу на конкретному сервері. Для його виявлення система аналізує кількість унікальних портів призначення, до яких звертається джерело протягом певного часового інтервалу.

Окремий тип правил пов’язаний з атаками методом грубої сили (англ. brute-force). Для такого сценарію характерна велика кількість невдалих спроб автентифікації, що можуть бути спрямовані на один обліковий запис, один сервіс або декілька різних вузлів. Джерелом таких подій можуть бути системні журнали операційної системи, журнали Virtual Private Network (VPN), журнали прикладних сервісів або інші джерела телеметрії. Важливо, що система має аналізувати не лише сам факт помилки входу, а й повторюваність, джерело підключення, цільовий сервіс і часовий інтервал.

Нормалізовані події надходять до механізму агрегації, на їх основі формуються метрики, після чого правила виявлення перевіряють наявність підозрілих шаблонів. Якщо умови правила виконуються, система створює інцидент і пов’язує його з подіями, які стали підставою для спрацювання.

Для збереження зв’язку між інцидентом і подіями використовується окрема логічна сутність incident events. Вона дозволяє прив’язати до одного інциденту декілька нормалізованих подій. Це важливо для розслідування, оскільки оператор може перейти від загального інциденту до конкретних записів телеметрії та перевірити, які саме події стали підставою для висновку системи.

У межах реалізації правила виявлення можна описувати через конфігураційний підхід. Це означає, що логіка виявлення не повинна бути жорстко прив’язана до одного конкретного джерела подій. Правило має описувати тип активності, набір умов, часовий інтервал, порогове значення, рівень критичності та дію, яку система може запропонувати після спрацювання. Такий підхід спрощує розширення системи новими правилами без повної зміни архітектури.

Приклад узагальненої структури правила виявлення можна подати у вигляді невеликого фрагмента конфігурації. Фактичний фрагмент із проєкту доцільно винести в додаток або замінити цим місцем після перевірки в репозиторії.

Лістинг 3.1 – Узагальнений приклад структури правила виявлення

```
      rule_key: repeated-blocked-access
      name: Repeated blocked access
      event_type: blocked_connection
      time_window_seconds: 300
      threshold: 10
      group_by:
        - source_ip
        - destination_port
      severity: medium
      confidence: 0.75
      description: Detects repeated blocked connection
attempts from one source.
```

Наведений приклад демонструє загальну логіку правила: система аналізує події певного типу в межах заданого часового вікна, групує їх за важливими полями та перевіряє, чи перевищено порогове значення. Якщо умови виконано, правило може створити інцидент відповідного типу. У реальній реалізації набір полів і параметрів може відрізнятися залежно від конкретного правила та джерела телеметрії.

Основні приклади правил виявлення, які відповідають логіці системи Castle Monitor, наведено в табл. 3.2.

Таблиця 3.2 – Приклади правил виявлення поведінкової активності

Правило	**Ознака активності**	**Джерела телеметрії**	**Можливий результат**
Repeated blocked access	Багаторазові заблоковані підключення з одного джерела	Firewall logs, syslog, normalized events	Інцидент повторних заборонених спроб доступу
Horizontal scan	Звернення одного джерела до багатьох вузлів	Мережеві події, firewall logs, агентські події	Інцидент горизонтального сканування
Vertical port scan	Звернення до багатьох портів одного вузла	Мережеві події, firewall logs	Інцидент сканування портів
Brute-force	Повторні невдалі спроби автентифікації	Linux logs, Windows events, VPN logs, service logs	Інцидент атаки методом грубої сили
IOC match	Збіг IP-адреси, домену або URL з індикатором компрометації	Threat intelligence, normalized events	Підвищення ризику або створення інциденту

Окремо слід зазначити роль індикаторів компрометації (англ. Indicators of Compromise, IoC). Такі індикатори можуть використовуватися як додатковий фактор під час аналізу подій. Наприклад, якщо IP-адреса джерела або домен збігається з відомим індикатором, система може підвищити рівень ризику або сформувати окремий інцидент. Водночас IOC не повинні розглядатися як єдина безумовна підстава для небезпечної автоматичної дії, оскільки зовнішні списки можуть містити застарілі або помилкові дані.

Під час інциденту система має враховувати рівень критичності та ступінь упевненості. Рівень критичності відображає потенційний вплив події або сценарію на інфраструктуру, а ступінь упевненості показує, наскільки надійними є ознаки виявленої активності. Наприклад, одинична подія може мати низьку впевненість, а повторюваний сценарій із декількома підтверджувальними ознаками — вищу. Такий підхід дозволяє пріоритезувати інциденти та не перевантажувати оператора великою кількістю незначних спрацювань.

Після створення інциденту він може бути відображений в інтерфейсі оператора (рис. 3.8). У такому поданні важливо показати не лише назву інциденту, а й час створення, статус, рівень критичності, джерело, ціль, короткий опис причини та пов’язані події. Це дозволяє оператору швидко оцінити ситуацію та прийняти рішення щодо подальшої перевірки або реагування.

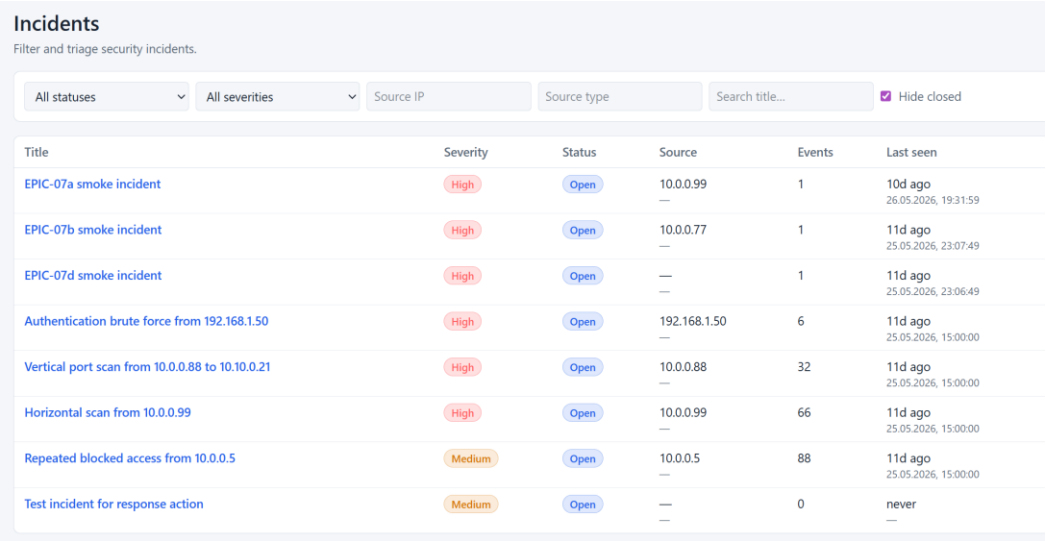


Рисунок 3.8 – Приклад відображення інцидентів в інтерфейсі оператора

З погляду STS-моделі сформований інцидент має бути пов’язаний не лише з конкретною подією, а й з рівнем інфраструктури, на якому проявилася активність. Така прив’язка дозволяє краще визначати зону потенційної компрометації та пов’язані сегменти.

Таким чином, реалізація правил виявлення та формування інцидентів у Castle Monitor побудована на послідовному перетворенні телеметрії в аналітичні об’єкти. Початкові події нормалізуються, на їх основі формуються метрики, правила виявлення перевіряють наявність поведінкових шаблонів, після чого створюються інциденти з посиланням на відповідні події. Такий підхід дозволяє системі виявляти не лише окремі підозрілі записи, а й повторювану активність, характерну для внутрішніх мережевих інцидентів.

3.5 Реалізація безпечного реагування та enforcement-механізму

У контексті моніторингу внутрішніх вразливостей локальної мережі є критично важливим етапом відповідне реагування на події, оскільки воно дозволяє зменшити потенційний радіус поширення загрози, обмежити активність підозрілого вузла або звернути увагу оператора на необхідність ручної перевірки. Водночас саме реагування створює додаткові ризики, оскільки помилкове блокування службової адреси, адміністративного доступу або критичного сервісу може негативно вплинути на роботу інфраструктури.

У зв’язку з цим у системі Castle Monitor механізм реагування реалізовано як контрольований процес, що складається з декількох послідовних етапів: формування інциденту, перевірка політик реагування, створення дії реагування, передавання дозволених директив агенту та отримання звіту про результат виконання або симуляції дії. Такий підхід дозволяє поєднати автоматизацію реагування з обмеженнями безпеки, необхідними для запобігання неконтрольованому впливу на мережеву інфраструктуру. Основою механізму реагування є політики реагування. Вони визначають, які дії система може планувати або виконувати у відповідь на певний тип інциденту.

Центральним елементом перевірки є механізм перевірки політик (англ. Policy Guard) (рис. 3.9). Він виконує роль проміжного захисного шару між інцидентом і дією реагування. Policy Guard перевіряє, чи дозволена запропонована дія, чи не належить ціль до переліку захищених об’єктів, чи не зачіпає дія критичні порти, чи відповідає вона поточному режиму виконання та чи не потребує підтвердження оператором. Якщо дія не проходить перевірку, вона не повинна виконуватися та не має потрапляти до набору директив для агента.

Рисунок 3.9 – Перегляд політик через меню Policies

Узагальнений приклад політики безпечного реагування наведено в лістингу 3.2. Він демонструє принцип, за яким небезпечна дія не виконується безпосередньо після формування інциденту, а проходить через додаткові обмеження: режим тестування (dry-run), підтвердження оператором, захищені порти та переліки адрес, які не можна блокувати автоматично.

Лістинг 3.2 – Узагальнений приклад політики безпечного реагування

```
policy_key: brute-force-approval
description: Brute-force response policy with operator approval
enabled: true
allowed_actions:
  - alert
  - suggest_block_ip
requires_approval: true
is_dry_run: true
ttl_seconds: 3600
protected_ports:
  - 22
  - 443
management_sources:
  - 10.0.0.0/24
never_block:
  - 127.0.0.1
  - 10.0.0.1
```

Після проходження перевірок Core API може сформувати набір директив реагування для агента. Такий набір директив доцільно розглядати як enforcement bundle (рис. 3.10), тобто контрольований пакет правил, який агент отримує від центрального компонента. Важливо, що Core API не повинен напряму змінювати локальні правила фільтрації на вузлі. Замість цього агент отримує лише дозволені директиви, перевіряє власний режим роботи та виконує або симулює дію локально.

На стороні агента локальне застосування правила залежить від операційної системи та доступних механізмів фільтрації. Для Linux-вузлів, зокрема Ubuntu, реагування може виконуватися через системні засоби фільтрації трафіку, такі як nftables або iptables. У тестовому середовищі система за замовчуванням використовує режим dry-run, що дозволяє перевірити запропоновану дію без зміни правил фільтрації трафіку. Водночас за умови ввімкнення live-режиму, наявності відповідних прав агента та успішного проходження перевірок Policy Guard система може застосовувати локальні правила блокування на вузлі.

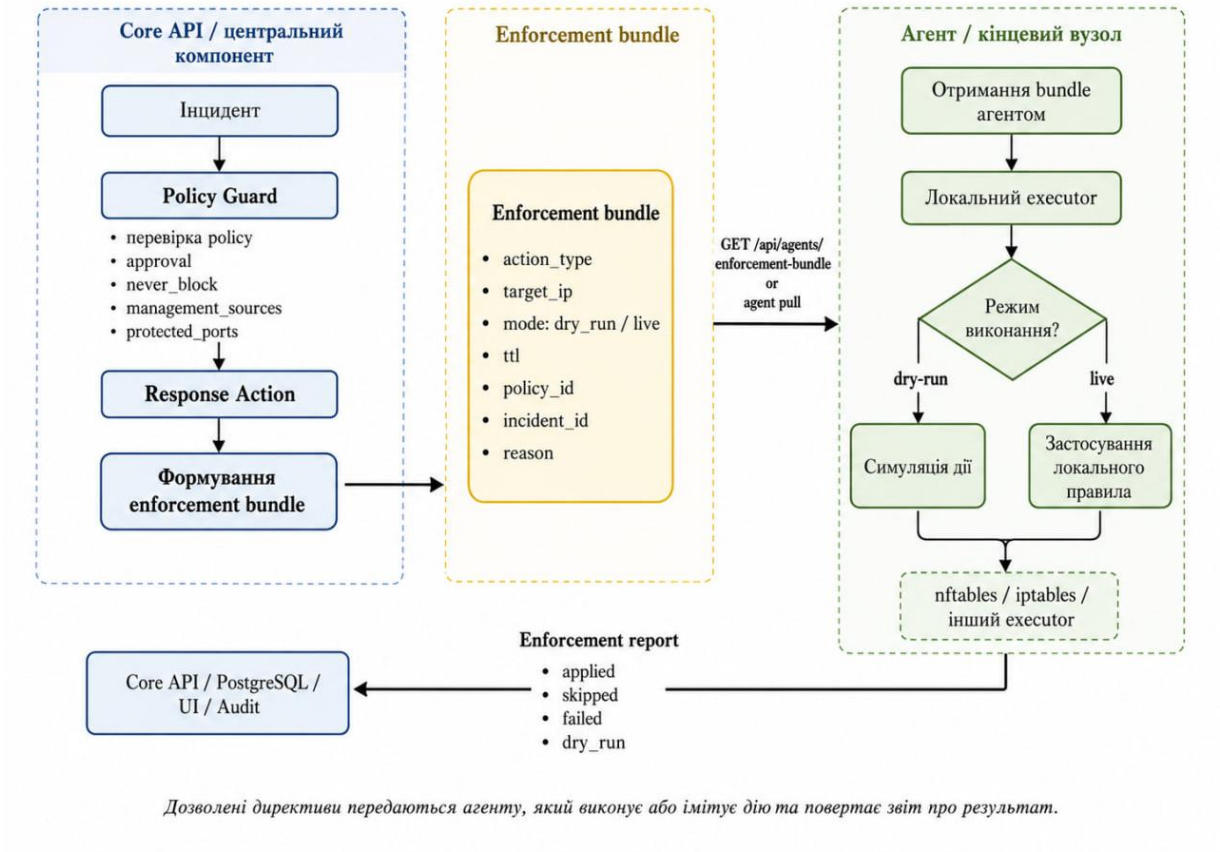


Рисунок 3.10 – Послідовність реагування через enforcement bundle

Аудит є обов’язковою складовою механізму реагування. Система має фіксувати створення політики, формування response action, рішення Policy Guard, підтвердження або відхилення дії, створення enforcement bundle і результат виконання агентом. Це важливо для подальшого розслідування, перевірки дій оператора, пояснення рішень системи та підтвердження того, що реагування виконувалося відповідно до визначених політик.

Основні принципи безпечного реагування в системі Castle Monitor наведено в табл. 3.3.

Таблиця 3.3 – Основні принципи безпечного реагування в Castle Monitor

Принцип	**Призначення**
Режим dry-run	Дозволяє перевірити запропоновану дію без фактичної зміни правил фільтрації
Підтвердження оператором	Забезпечує контроль людини перед виконанням дій, які можуть вплинути на доступність
Never block	Захищає критичні адреси або мережі від автоматичного блокування
Management sources	Запобігає втраті адміністративного доступу до інфраструктури
Protected ports	Захищає критичні службові порти від необережного блокування
Enforcement bundle	Передає агенту лише дозволені директиви реагування
Аудит	Фіксує рішення системи, дії оператора та результат виконання

Узагальнено логіку реагування можна подати як послідовність: інцидент передається до механізму реагування, Policy Guard перевіряє політики й винятки, система створює response action у дозволеному статусі, агент отримує enforcement bundle, виконує або симулює дію та повертає звіт. Якщо будь-який етап перевірки не виконано, дія має бути відхилена, пропущена або залишена в безпечному стані.

У розробленій системі реалізовано та продемонстровано механізм автоматизованого реагування на інциденти на основі політик безпеки. Система здатна не лише сформувати інцидент, а й підготувати або виконати дію локалізації через агентський механізм.

3.6 Експериментальна перевірка системи в тестовому середовищі

Для підтвердження працездатності запропонованих проєктних рішень було виконано експериментальну перевірку системи Castle Monitor у контрольованому тестовому середовищі. Метою перевірки було встановити, чи здатна система виконувати основні функції, визначені в попередніх розділах: розгортання серверних компонентів, підключення агентів, приймання телеметрії, збереження подій, формування інцидентів, відображення результатів в інтерфейсі оператора та підготовку дій реагування на основі політик.

Тестовий стенд було побудовано як локальне середовище з використанням контейнеризованих серверних компонентів і окремих вузлів для перевірки агентської частини (рис. 3.11). Контейнеризація дозволила відокремити основні сервіси системи один від одного, спростити повторне розгортання та забезпечити відтворюваність експериментальної перевірки. До складу серверної частини входили Core API, база даних PostgreSQL, frontend-інтерфейс оператора, Grafana та компонент приймання syslog-подій.

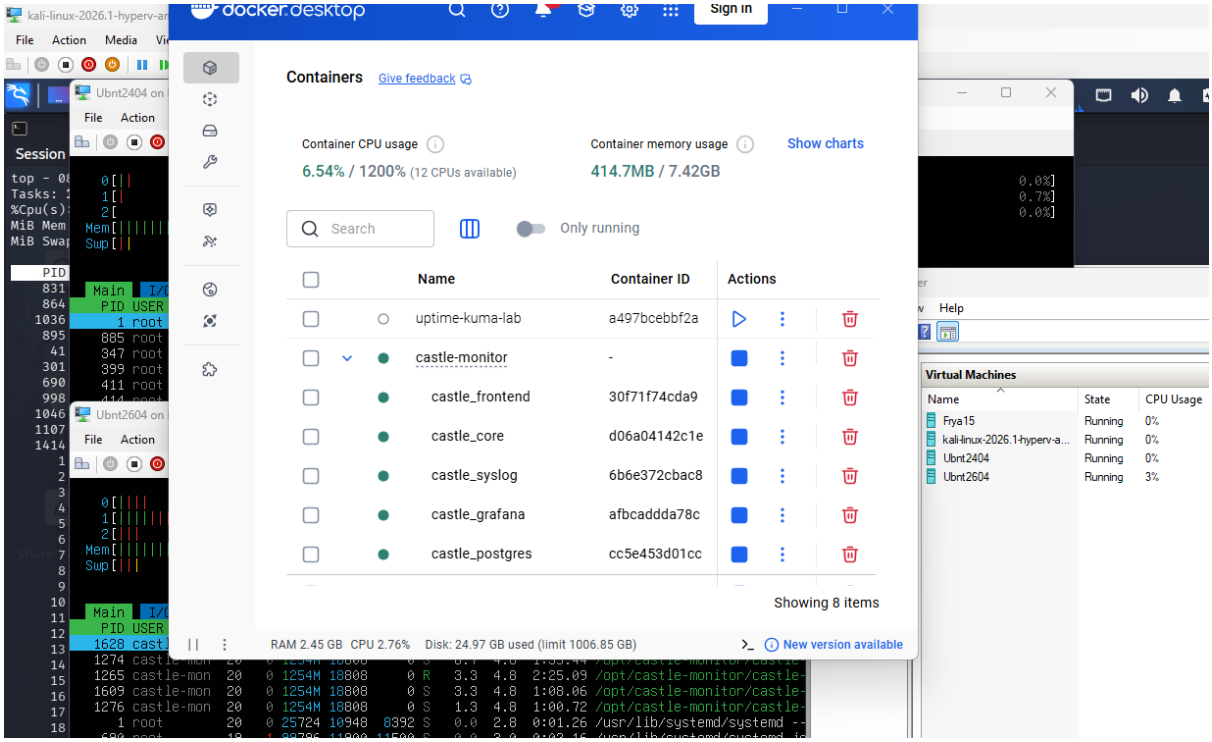


Рисунок 3.11 – Тестовий стенд для Castle Monitor

Фактична перевірка показала, що серверна частина системи може бути розгорнута в межах одного тестового вузла. У режимі простою контейнери Core API, PostgreSQL, frontend, Grafana та syslog-компонент не створювали значного навантаження на обчислювальні ресурси. Це підтверджує можливість використання системи в навчальному або тестовому середовищі без потреби у виділенні потужного серверного обладнання на початковому етапі. Водночас такі показники не слід розглядати як остаточну характеристику продуктивності для промислового середовища.

Для перевірки агентської частини було використано вузли з операційними системами Ubuntu/Linux, Windows і FreeBSD. На них перевірялися встановлення або запуск агента, налаштування конфігурації, підключення до Core API, процедура реєстрації, підтвердження агента через інтерфейс оператора, heartbeat-повідомлення та передавання службової інформації. Найбільш повно було опрацьовано сценарій Ubuntu/Linux, оскільки саме для нього також перевірялися отримання enforcement bundle, dry-run режим і контрольована можливість локального реагування через системні засоби фільтрації трафіку.

Експериментальна перевірка була спрямована не лише на запуск компонентів системи, а й на підтвердження того, як система відпрацьовує типові сценарії виявлення, локалізації та реагування. Окремо перевірялася робота агентів на різних операційних системах, зокрема Ubuntu/Linux, Windows і FreeBSD, а також можливість отримання телеметрії від безагентних джерел через syslog і Simple Network Management Protocol (SNMP). При цьому основний акцент перевірки було зроблено не на самому факті запуску агента, а на тому, як отримані події використовуються для формування інцидентів, перевірки політик і створення дій реагування.

Результати експериментальної перевірки сценаріїв виявлення та реагування наведено в табл. 3.4.

Таблиця 3.4 – Результати експериментальної перевірки сценаріїв виявлення та реагування системою Castle Monitor

Сценарій	**Джерела перевірки**	**Результат**
Підключення агентів	Ubuntu, Windows, FreeBSD	Виконано
Приймання телеметрії	агенти, syslog/SNMP	Виконано
Вертикальне сканування	normalized events / firewall logs	Інцидент сформовано
Горизонтальне сканування	normalized events / firewall logs	Інцидент сформовано
Brute-force	Linux/Windows/service logs	Інцидент сформовано
Lateral movement / east-west	мережеві події	Інцидент сформовано
C2-подібні регулярні з’єднання	повторювані flow-події	Інцидент сформовано
Policy Guard / dry-run / approval	response policies	Дія реагування сформована без неконтрольованого блокування

Окремо для демонстрації можливості роботи системи з малоресурсними пристроями було використано мінікомп’ютер Raspberry Pi 3B з операційною системою «Ubuntu 26.04». Такий пристрій не є класичним сервером або робочою станцією, однак за своїми характеристиками наближений до класу периферійних або IoT-подібних вузлів, які можуть використовуватися в локальних мережах для виконання допоміжних функцій, збору даних або роботи з периферійним обладнанням.

Використання Raspberry Pi 3B у тестовому середовищі (рис. 3.12) демонструє, що агентська архітектура Castle Monitor може бути застосована не лише на повноцінних серверних або користувацьких системах, а й на пристроях з обмеженими обчислювальними ресурсами. Це розширює потенційну сферу використання системи для середовищ, у яких присутні малоресурсні вузли, контролери, шлюзи, пристрої Internet of Things (IoT) або допоміжні

інфраструктурні компоненти. У контексті STS-моделі такі пристрої можуть належати до рівня Ground / «земля», оскільки вони часто пов’язані з фізичною або навколофізичною частиною інфраструктури.

Рисунок 3.12 – Використання Raspberry Pi 3B як малоресурсного вузла в тестовому середовищі Castle Monitor

У межах експериментальної перевірки було також перевірено в тестовому режимі, dry-run. Він дозволяє сформувати дію реагування без фактичної зміни правил фільтрації трафіку. Такий режим є важливим для контрольованого тестування політик, оскільки дає змогу оцінити, яку дію система запропонувала б виконати, не створюючи ризику блокування критичних ресурсів. Dry-run у цьому випадку розглядається не як відсутність реагування, а як безпечний етап перевірки перед можливим застосуванням правила, залишаючи повний функціонал моніторингу (рис. 3.13).

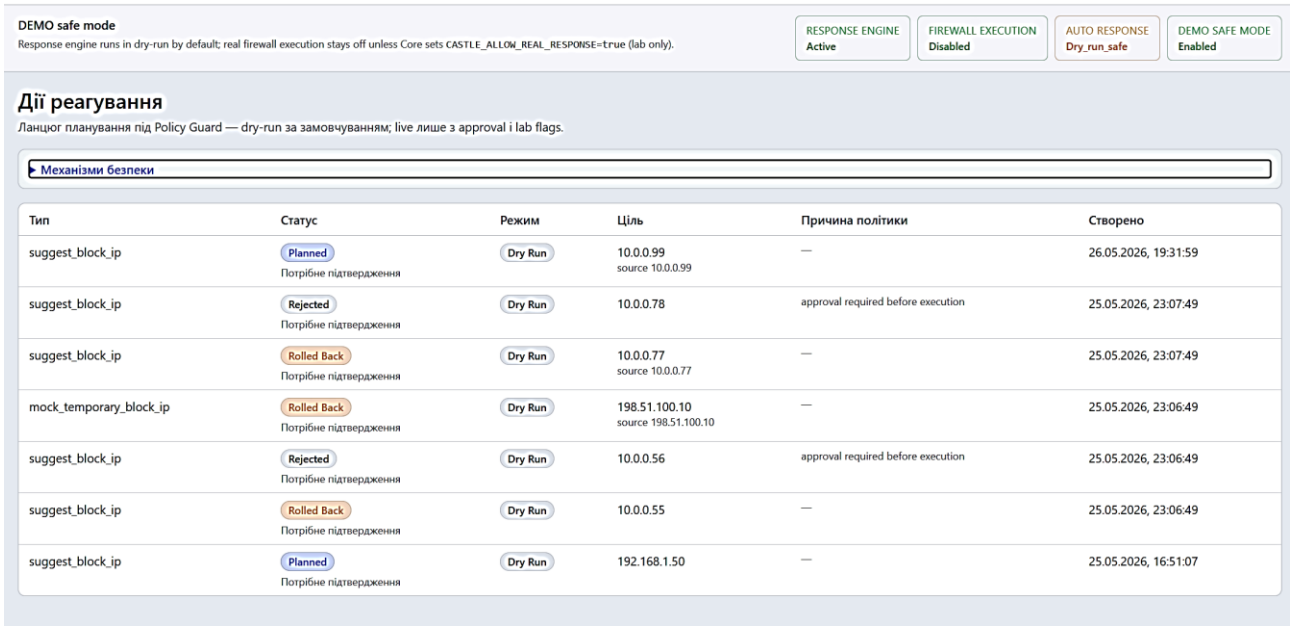


Рисунок 3.13 – Приклад дії реагування в режимі dry-run

Під час перевірки було підтверджено, що система може використовуватися як додатковий рівень контролю внутрішньої мережевої активності. Вона не лише збирає події, а й забезпечує зв’язок між телеметрією, інцидентами, політиками та діями реагування. Це відповідає меті роботи, оскільки запропонований підхід спрямований на підвищення ефективності виявлення, локалізації та реагування на внутрішні мережеві інциденти.

3.7 Обмеження реалізації та напрями подальшого розвитку

Проведена реалізація та експериментальна перевірка системи Castle Monitor підтвердили працездатність основних компонентів. Водночас поточна реалізація має низку обмежень, які необхідно враховувати під час подальшого розвитку системи.

Першим обмеженням є те, що 5-tuple контекст, який включає source_ip, destination_ip, source_port, destination_port і protocol, використовується переважно для опису подій, інцидентів і мережевої взаємодії. Повноцінне автоматичне застосування правил міжмережевого екрана за всіма п’ятьма параметрами потоку не розглядається як завершена частина поточної реалізації. На цьому етапі enforcement-механізм переважно орієнтований на адресне блокування або інші

доступні параметри, а розширення до повноцінних 5-tuple правил може бути винесене до подальшого розвитку.

Другим обмеженням є залежність live enforcement від операційної системи, прав процесу агента та доступності локальних засобів фільтрації трафіку. Для Linux/Ubuntu контрольоване застосування правил може виконуватися за умови наявності відповідних прав, live-режиму та засобів nftables або iptables. Для Windows і FreeBSD фактичне застосування правил залежить від підтримуваного executor, прав агента та особливостей локального механізму фільтрації.

Третім обмеженням є залежність якості виявлення від повноти та якості телеметрії. Якщо джерело не передає необхідні поля або журнали мають неповний формат, система може сформувати менш точний контекст інциденту. Тому для практичного використання важливими залишаються коректна інвентаризація джерел, налаштування агентів, актуальність правил нормалізації та повнота опису сегментів інфраструктури.

До напрямів подальшого розвитку системи можна віднести розширення підтримки 5-tuple правил реагування, покращення роботи з Windows Event Log і FreeBSD-журналами, впровадження механізму самооновлення агента та розширення risk scoring із поступовим зменшенням ризику за часом. Також перспективним є розвиток візуалізації інцидентів, політик реагування та зон потенційної компрометації в інтерфейсі оператора.

Окремим напрямом подальшого розвитку може бути впровадження модуля на основі штучного інтелекту, який допомагатиме оператору або адміністратору формувати YAML-сценарії виявлення та реагування на основі опису інциденту природною мовою. Такий модуль міг би аналізувати історію подій, інциденти, дії реагування та політики безпеки, після чого пропонувати нові правила виявлення, шаблони реагування або рекомендації щодо локалізації загрози. Водночас такі рекомендації не повинні виконуватися автоматично без перевірки: сформовані штучним інтелектом правила мають проходити валідацію, тестування в режимі dry-run і підтвердження оператором перед використанням у продуктивному

середовищі.

Таким чином, поточна реалізація Castle Monitor створює працездатну основу для моніторингу, поведінкового аналізу та контрольованого реагування на внутрішні мережеві інциденти. Виявлені обмеження не зменшують практичної цінності системи, а визначають напрями її подальшого вдосконалення.

Висновки до розділу 3

У третьому розділі було описано практичну реалізацію та експериментальну перевірку системи. У межах реалізації було забезпечено централізований збір і зберігання телеметрії, роботу з агентами, нормалізацію подій, формування метрик, створення інцидентів і підготовку дій реагування.

Експериментальна перевірка в тестовому середовищі підтвердила працездатність основних функцій системи. Було перевірено розгортання серверної частини, основні функції агентів Ubuntu/Linux, Windows і FreeBSD. Також підтверджено відпрацювання політик реагування, зокрема dry-run, approval, never_block, management_sources і protected_ports.

Реалізований механізм реагування побудовано за принципом контрольованого застосування дій: система перевіряє дію через Policy Guard, враховує політики безпеки, режим виконання, критичні адреси, службові порти та потребу в підтвердженні оператора. Таким чином, результати розділу підтверджують, що Castle Monitor здатна виконувати функції збору телеметрії, поведінкового аналізу, локалізації інцидентів і контрольованого реагування, а визначені обмеження щодо повного 5-tuple enforcement і залежності live-режиму від операційної системи формують напрями подальшого розвитку системи.

ВИСНОВКИ

В результаті виконання кваліфікаційної бакалаврської роботи було досягнуто поставленої мети — підвищено ефективність виявлення, локалізації та реагування на внутрішні мережеві інциденти шляхом застосування поведінкового аналізу шаблонної активності зловмисників і розроблення системи моніторингу внутрішніх вразливостей локальної мережі Castle Monitor.

У ході роботи було досліджено сучасні засоби діагностики, захисту та моніторингу локальних мереж, зокрема системи моніторингу, централізованого збору журналів, аналізу подій безпеки, системи виявлення вторгнень та засоби мережевого контролю. Було розглянуто можливості таких рішень, як Nagios, Elastic Stack, Graylog, Snort, Wazuh та інші інструменти, визначено їх переваги й обмеження в контексті виявлення внутрішніх мережевих інцидентів.

У роботі запропоновано та описано Structure-Based Telemetry Source Model (STS), яка використовується для структуризації джерел телеметрії відповідно до рівня їх походження в інфраструктурі. Запропонована модель дозволяє не лише класифікувати події, а й визначати потенційну зону недовіри, рівень походження інциденту та можливий радіус поширення загрози в локальній мережі.

У межах практичної реалізації було створено основні компоненти системи: серверну частину, інформаційну модель бази даних, механізм роботи агентів, механізми збору та нормалізації подій, правила виявлення поведінкових аномалій, формування інцидентів і створення дій реагування. Система підтримує агентські й безагентні джерела телеметрії, що дозволяє використовувати її для моніторингу різних типів вузлів і мережевих пристроїв.

Реалізовані механізми виявлення дозволяють аналізувати шаблонну активність зловмисників, зокрема вертикальне та горизонтальне сканування, brute-force атаки, repeated blocked access, аномальний east-west трафік, бокове переміщення між вузлами та C2-подібні регулярні з’єднання. Для цього система використовує нормалізовану телеметрію, часові вікна, поведінкові метрики та інфраструктурний контекст STS-моделі.

Окрему увагу в роботі приділено безпечному реагуванню на інциденти. Реагування реалізовано через політики безпеки, Policy Guard, режим dry-run, підтвердження оператором, механізми never_block, management_sources, protected_ports, response actions та enforcement bundle для агентів. Такий підхід дозволяє не лише формувати інциденти, а й контрольовано готувати або виконувати дії локалізації без ризику некерованого блокування критичних ресурсів.

Експериментальна перевірка в тестовому середовищі підтвердила працездатність основних компонентів Castle Monitor. Додатково було використано Raspberry Pi 3B для демонстрації можливості роботи системи з малоресурсними пристроями та потенційного застосування в середовищах з Internet of Things (IoT)-подібними вузлами.

Розроблену систему доцільно використовувати як додатковий рівень контролю внутрішньої мережевої активності в корпоративних, навчальних, дослідницьких та критичних інфраструктурах, де важливими є своєчасне виявлення підозрілої поведінки, локалізація потенційно скомпрометованих вузлів і контрольоване реагування на інциденти.

Подальший розвиток системи може бути спрямований на розширення підтримки 5-tuple правил реагування, покращення роботи з журналами, впровадження самооновлення агентів, покращення підрахунку ризиків та створення модуля на основі штучного інтелекту для допомоги оператору у формуванні YAML-сценаріїв виявлення й реагування.

Таким чином, результати роботи підтверджують можливість побудови системи моніторингу внутрішніх вразливостей локальної мережі, яка поєднує збір телеметрії, поведінковий аналіз, структурну локалізацію інцидентів і контрольоване реагування на дії зловмисників.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Волков В. В., Журавська І. М. Метод локалізації мережевих інцидентів на основі моделі STS. *Ольвійський форум – 2026: Стратегії країн Причорноморського регіону в геополітичному просторі* : тези доп. XXIII Міжнар. наук. конф., Миколаїв, 29 черв.–04 лип. 2026 р. Миколаїв : ЧНУ ім. Петра Могили, 2026. *(прийнято до друку).*

2. Волков В. В. Система моніторингу внутрішніх вразливостей локальної мережі на основі аналізу шаблонної активності зловмисників. *Free and Open Source Software (FOSS'2026)* : тези доп. XVII Міжнар. наук.-практ. конф. Харків, 11–12 лютого 2026 р. Харків : ХНЕУ ім. Семена Кузнеця, 2026. С. 86–88. URL: https://foss.kn-it.info/uploads/foss-2026-theses.pdf (дата звернення: 12.05.2026).

3. Kaklauskas L., Pilypas I. Solution of remote management system for computer network infrastructure with integrated NetBox. *Professional studies: theory and practice*. 2025. Vol. 29(1). P. 4–12. DOI: 10.56131/pstp.2025.29.1.347.

4. The premier network source of truth. URL: https://netboxlabs.com/docs/netbox/ (Last accessed:12.02.2026).

5. Smiliotopoulos C., Shiaeles S., Kolokotronis N. Detecting lateral movement: A systematic survey. *Journal of Network and Computer Applications*. 2024. Vol. 223. DOI: 10.1016/j.heliyon.2024.e26317.

6. Bowman B., Laprade C., Ji H., Howie Huang H. Detecting lateral movement in enterprise computer networks using unsupervised graph learning. *RAID 2020* : Proc. of the 23rd International Symposium on Research in Attacks, Intrusions and Defenses. 2020. P. 257–268. URL: https://www.usenix.org/system/files/raid20-bowman.pdf (Last accessed: 13.05.2026).

7. Papadogiannaki E., Tsirantonakis G., Ioannidis S. Network intrusion detection in encrypted traffic. *2022 IEEE Conference on Dependable and Secure Computing (DSC)*. 2022. DOI: 10.1109/DSC54232.2022.9888942.

8. ENISA. Encrypted traffic analysis. *European Union Agency for Cybersecurity*, 2019. URL:

https://www.enisa.europa.eu/sites/default/files/publications/ENISA%20Report%20-%20Encrypted%20Traffic%20Analysis.pdf (Last accessed: 13.05.2026).

9. Herranz-Oliveros D., García-Teodoro P. Unsupervised learning for lateral-movement-based threat detection. *Electronics.* 2024. Vol. 13 (19). DOI: 10.3390/electronics13193944.

10. Chen P.-Y., Choudhury S., Rodriguez L., Hero A., Ray I. Enterprise cyber resiliency against lateral movement: A graph theoretic approach. *arXiv*. 2019. URL: https://arxiv.org/abs/1905.01002 (Last accessed: 13.05.2026).

11. Benmalek M., Djoudi M., Harous S. Ransomware on cyber-physical systems: Taxonomies, analysis and mitigation strategies. *Internet of Things and Cyber-Physical Systems*. 2024. Vol. 4. P. 1–20. DOI: 10.1016/j.iotcps.2023.12.001.

12. Wickramasinghe N., Shaghaghi A., Tsudik G., Jha S. SoK: Decoding the Enigma of encrypted network traffic classifiers. *arXiv.* 2025. URL: https://arxiv.org/abs/2503.20093 (Last accessed: 13.05.2026).

13. Sharma A., Singh K. A survey on encrypted network traffic: A comprehensive review of classification techniques and challenges. *Computer Networks*. 2025. DOI: 10.1016/j.comnet.2024.110984

14. Akibis M., Pereira J., Clark D., Show D., Alvarez H. Measuring ransomware propagation patterns via network traffic analysis: An automated approach : preprint. 2024. 11 p. DOI: 10.21203/rs.3.rs-5180048/v1.

15. Radivilova T., Kirichenko L., Ageyev D., Tawalbeh M., Bulakh V. Decrypting SSL/TLS traffic for hidden threats detection. *arXiv.* 2019. URL: https://arxiv.org/abs/1904.08383 (Last accessed: 13.05.2026).

16. Абрамов В. О., Глушак О. М., Плоха А. Ю., Довженко Т. П. Проєктування мережевої інфраструктури з урахуванням вимог кібербезпеки: підходи та реалізація на базі Cisco. *Кібербезпека: освіта, наука, техніка.* 2025. № 4 (28). С. 59–72. DOI 10.28925/2663-4023.2025.29.845.

17. Microsoft. WannaCry ransomware attack lessons learned. URL https://www.microsoft.com/en-us/microsoft-cloud/blog/healthcare/2017/06/06/

wannacry-ransomware-attack-lessons-learned/ (Last accessed: 13.05.2026).

18. Snort documentation. URL: https://docs.snort.org/rules/ (Last accessed: 13.05.2026).

19. Suricata documentation. URL: https://docs.suricata.io/en/suricata-8.0.4/ (Last accessed: 13.05.2026).

20. Elastic stack documentation. URL: https://www.elastic.co/docs (Last accessed: 13.05.2026).

21. Wazuh documentation. URL: https://documentation.wazuh.com (Last accessed: 13.05.2026).

22. Graylog. Centralized log management. URL: https://graylog.org/use-cases/centralized-log-management/ (Last accessed: 13.05.2026).

23. Announcing Prometheus 3.0. URL: https://prometheus.io/blog/2024/11/14/prometheus-3-0/ (Last accessed: 13.05.2026).

24. Elastic Stack – IIS (Internet Information Services) integration. URL: https://www.elastic.co/docs/reference/integrations/iis (дата звернення: 13.05.2026).

25. What is Syslog and how does it work? URL: https://graylog.org/post/what-is-syslog-and-how-does-it-work/ (Last accessed: 13.05.2026).

26. Snort alerts in pfSense. URL: https://docs.netgate.com/pfsense/en/latest/packages/snort/alerts.html (Last accessed: 13.05.2026).

27. Bondhala S. Modern defense paradigms: Zero Trust architecture, network segmentation, and micro-segmentation. *International Journal of Scientific Research in Computer Science, Engineering and Information Technology*. 2025, Vol. 11 (2). P. 2230–2239. DOI: 10.32628/CSEIT25112714

ДОДАТОК А

Матеріали апробації роботи

А.1 XXIII Міжнародна наукова конференція «Ольвійський форум – 2026: стратегії країн Причорноморського регіону в геополітичному просторі»

Міністерство освіти та науки України
Чорноморський національний університет імені Петра Могили
Первинна профспілкова організація ЧНУ імені Петра Могили

XXIII Міжнародна наукова конференція
«ОЛЬВІЙСЬКИЙ ФОРУМ-2026: СТРАТЕГІЇ КРАЇН ПРИЧОРНОМОРСЬКОГО РЕГІОНУ В ГЕОПОЛІТИЧНОМУ ПРОСТОРІ»,
з 29 червня по 04 липня 2026 року

СЕКЦІЯ: ТЕХНІЧНІ НАУКИ
ПІДСЕКЦІЯ: Комп'ютерна інженерія

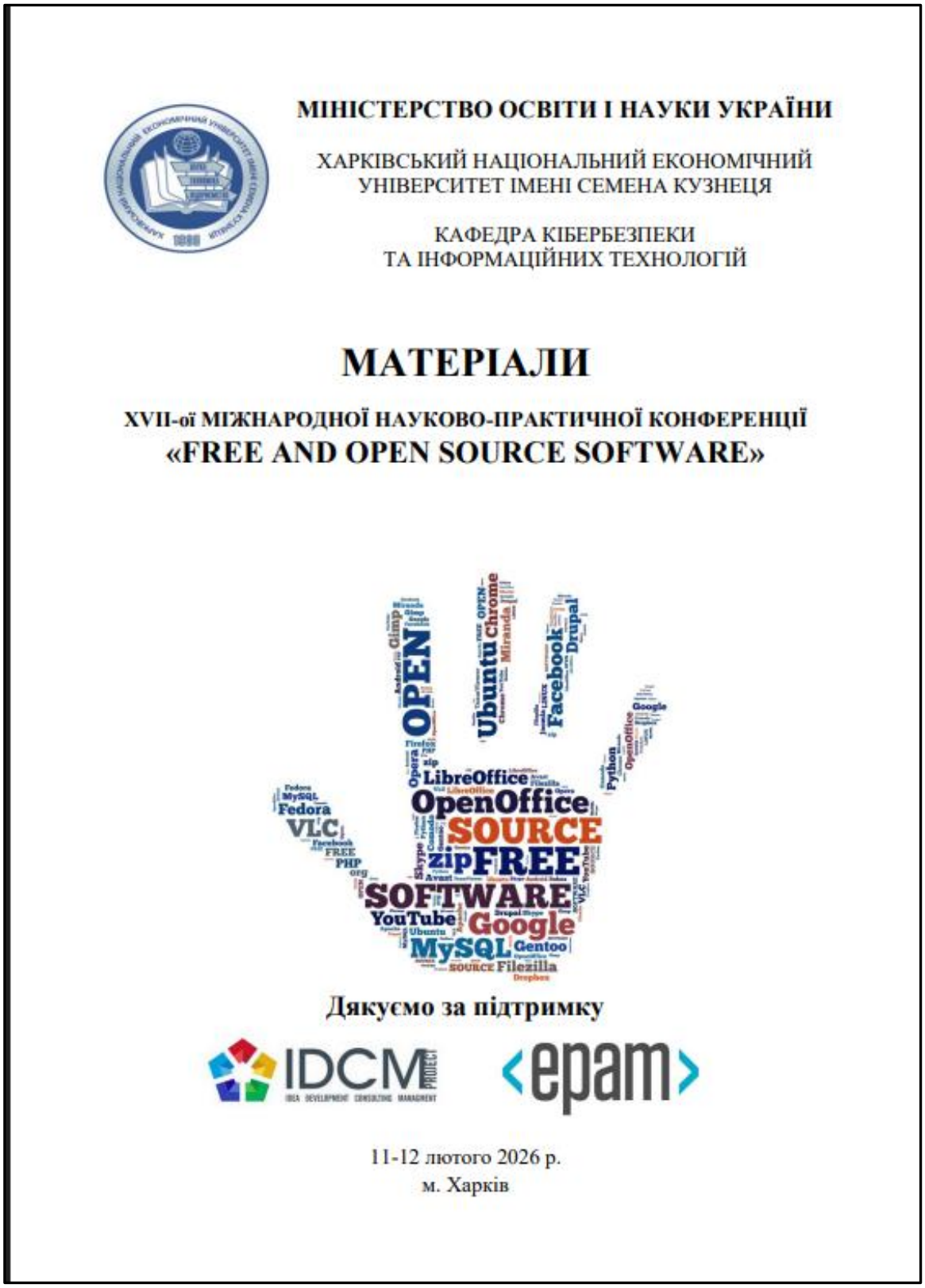
Дата, час проведення: 03.07.2026 о 14:00
Zoom-посилання: https://us02web.zoom.us/j/88140506050?pwd=UINkAjIhaVYGbCb8bEFGU7aRnLgjAR.1

Керівник підсекції: **Савінов В. Ю.** – канд. техн. наук, доцент кафедри комп'ютерної інженерії
Секретар підсекції: **Афонін Ю. С.** – аспірант кафедри комп'ютерної інженерії
Мета проведення: обмін науковими поглядами щодо перспектив розвитку комп'ютерної інженерії в Україні та обговорення перспективних розробок.

Перелік доповідей:

1. **Чуйко Г. П.** (д-р фіз.-мат. наук, професор, професор кафедри комп'ютерної інженерії), **Дарнапук Є. С.** (PhD, доцент (б. в. з.) кафедри комп'ютерної інженерії), **Кравченко П. К.** (магістрантка кафедри комп'ютерної інженерії, Чорноморський нац. ун-т ім. Петра Могили, м. Миколаїв, Україна). Українська квантова освіта в глобальному контексті.
2. **Афонін Ю. С.** (аспірант кафедри комп'ютерної інженерії), **Яковіщин А. Я.** (магістрант кафедри АКІТ), **Савінов В. Ю.** (канд. техн. наук, доцент, доцент кафедри комп'ютерної інженерії, Чорноморський нац. ун-т ім. Петра Могили, м. Миколаїв, Україна). Концептуальні засади поєднання гетерогенних роїв безпілотних апаратів та машинного навчання для вирішення завдань гуманітарного розмінування.
3. **Гончаров Д. С.** (аспірант кафедри комп'ютерної інженерії, Чорноморський нац. ун-т ім. Петра Могили, м. Миколаїв, Україна). **Кандиба І. О.** (PhD, ст. викладач кафедри інженерії програмного забезпечення). Порівняння монолітної та багаторівневої архітектур систем моніторингу стану здоров'я людини.
4. **Худолій Є. П.** (аспірант), **Дарнапук Є. С.** (PhD, доцент (б. в. з.) кафедри комп'ютерної інженерії, Чорноморський нац. ун-т ім. Петра Могили, м. Миколаїв, Україна). Порівняльний аналіз архітектурних підходів до крос-доменної верифікації ідентичності в умовах нульової довіри у гетерогенних багатохмарних середовищах.
5. **Шевченко В. В.** (бакалавр), **Дарнапук Є. С.** (PhD, доцент кафедри комп'ютерної інженерії, Чорноморський нац. ун-т ім. Петра Могили, м. Миколаїв, Україна). Проектування розподіленої PACS-системи для обміну DICOM-зображеннями з використанням Raspberry Pi 4.
6. **Чередниченко Д. О.** (бакалавр), **Дарнапук Є. С.** (PhD, доцент кафедри комп'ютерної інженерії, Чорноморський нац. ун-т ім. Петра Могили, м. Миколаїв, Україна). Методи оцінювання стресового стану за допомогою IoT-системи на базі ESP8266 V3.0 CH340.
7. **Бурлаченко І. С.** (старший викладач кафедри комп'ютерної інженерії, Чорноморський нац. ун-т ім. Петра Могили, м. Миколаїв, Україна). Особливості мультиагентних систем розпізнавання жестів на базі ESP32.
8. **Крайник Я. М.** (канд. техн. наук, доцент, доцент кафедри комп'ютерної інженерії, Чорноморський нац. ун-т ім. Петра Могили, м. Миколаїв, Україна). HLS Implementation of the QOI compression for FPGA.
9. **Волков В. В.** (бакалавр), **Журавська І. М.** (д-р техн. наук, проф., завідувач кафедри комп'ютерної інженерії, Чорноморський нац. ун-т ім. Петра Могили, м. Миколаїв, Україна). Метод локалізації мережевих інцидентів на основі моделі STS.
10. **Салтовський Б. Г.** (старший викладач кафедри комп'ютерної інженерії, Чорноморський нац. ун-т ім. Петра

А.2 XVII Міжнародна науково-практична конференція «Free and Open Source Software» (FOSS'2026)



МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ ЕКОНОМІЧНИЙ УНІВЕРСИТЕТ ІМЕНІ СЕМЕНА КУЗНЕЦЯ

КАФЕДРА КІБЕРБЕЗПЕКИ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

МАТЕРІАЛИ

XVII-ої МІЖНАРОДНОЇ НАУКОВО-ПРАКТИЧНОЇ КОНФЕРЕНЦІЇ
«FREE AND OPEN SOURCE SOFTWARE»

Дякуємо за підтримку

IDCM PROTECH

epam

11-12 лютого 2026 р.
м. Харків

СИСТЕМА МОНІТОРИНГУ ВНУТРІШНІХ ВРАЗЛИВОСТЕЙ ЛОКАЛЬНОЇ МЕРЕЖІ НА ОСНОВІ АНАЛІЗУ ШАБЛОННОЇ АКТИВНОСТІ ЗЛОВМИСНИКІВ

Волков В.В.
E-mail: vikvolk99@gmail.com
Миколаїв, Чорноморський національний університет імені Петра Могили

У сучасних умовах значна кількість компаній експлуатує великі парки персональних комп’ютерів, серверів та різноманітних кінцевих пристроїв, які мають бути налаштовані з урахуванням різних сценаріїв використання та підвищених вимог до інформаційної безпеки. [1] Паралельно з цим багато сервісів використовують розгалужену інфраструктуру віртуальних машин, зокрема в демілітаризованих зонах (DMZ), що суттєво ускладнює контроль стану мережі.)[2] [3]

Попри критичну важливість таких сегментів, [4] системам моніторингу внутрішніх мереж та виявленню потенційних вразливостей часто приділяється недостатньо уваги. [5] Водночас інженеру, який відповідає за експлуатацію та безпеку інфраструктури, необхідно мати актуальну та цілісну картину стану мережі для своєчасного виявлення інцидентів і реагування на них.

У межах даної роботи розглядається підхід до побудови системи моніторингу та виявлення внутрішніх вразливостей локальних мереж, що базується на використанні шаблонів поведінкової активності потенційного зловмисника. [6] [7]Запропонований підхід дозволяє оперативно ідентифікувати підозрілу активність, швидко знаходити скомпрометовані або потенційно небезпечні хости та ізолювати їх без суттєвого впливу на роботу всієї мережі.

Наукова новизна роботи полягає у:

– застосуванні поведінкових шаблонів для виявлення внутрішніх вразливостей локальної мережі без використання глибокого аналізу пакетів (DPI); [8]

– поєднанні централізованої аналітики з децентралізованим виконанням політик реагування;

– адаптації механізмів реагування відповідно до ролі та контексту кінцевого хоста в мережі.

Практична цінність роботи:

– можливість впровадження системи у гетерогенних середовищах (Linux, Windows, FreeBSD);

– низькі вимоги до обчислювальних ресурсів агентів;

– інтеграція з існуючими системами моніторингу та візуалізації, зокрема Grafana;

– можливість автоматичного створення правил міжмережевих екранів для швидкого реагування, а також потенційне керування blacklist-ами на мережевому маршрутизаторі.

На представленій схемі (рис. 1) зображено узагальнену архітектуру системи моніторингу та виявлення внутрішніх вразливостей локальної мережі з акцентом на безпечну взаємодію компонентів і розмежування зон доступу.

Взаємодія з системою здійснюється оператором, який підключається до графічного інтерфейсу управління. Для забезпечення конфіденційності та цілісності переданих даних доступ оператора можливий виключно з використанням актуальних версій протоколу TLS, що знижує ризик перехоплення або підміни трафіку під час керування системою та перегляду результатів моніторингу.

На представленій схемі зображено узагальнену архітектуру системи моніторингу та виявлення внутрішніх вразливостей локальної мережі з акцентом на безпечну взаємодію компонентів і розмежування зон доступу.

Взаємодія з системою здійснюється оператором, який підключається до графічного інтерфейсу управління. Для забезпечення конфіденційності та цілісності переданих даних доступ оператора можливий виключно з використанням актуальних версій протоколу TLS,

що знижує ризик перехоплення або підміни трафіку під час керування системою та перегляду результатів моніторингу.

Рисноук 1 – Архітектура системи

Що стосується взаємодії системи, вона представлена як подієво-орієнтований процес між децентралізованими точками моніторингу та центральним ядром аналізу. У процесі роботи агенти фіксують мережеву активність хостів і передають узагальнені метадані до Core-компонента, де відбувається кореляція подій та ідентифікація поведінкових шаблонів. У разі виявлення активності, що відповідає формалізованим ознакам атаки, система виконує оцінку рівня ризику з урахуванням контексту та історії взаємодій. На основі результатів оцінювання формується керуюче рішення, яке може передбачати обмеження або блокування мережевої взаємодії. Прийняті рішення централізовано зберігаються та використовуються для подальшої кореляції подій іншими компонентами системи. Реалізація реагування здійснюється на рівні операційної системи хоста шляхом автоматизованого застосування правил безпеки.

Таким чином, взаємодія компонентів утворює замкнений цикл, що охоплює збір даних, аналіз, оцінку ризиків та автоматизоване реагування, забезпечуючи своєчасне виявлення та нейтралізацію внутрішніх загроз у локальній мережі.

Розроблена система моніторингу та виявлення внутрішніх вразливостей локальної мережі побудована за модульним принципом і складається з логічно відокремлених компонентів, що взаємодіють між собою через захищені програмні інтерфейси. Такий підхід забезпечує масштабованість, гнучкість і можливість адаптації системи до різних інфраструктурних середовищ.

Frontend призначений для взаємодії оператора з системою та реалізує функції керування, контролю й візуалізації. Через веб-інтерфейс оператор отримує доступ до налаштування конфігурацій, політик безпеки та сценаріїв реагування, а також до перегляду подій безпеки, результатів аналізу та історії дій системи. Взаємодія з ядром здійснюється виключно через захищені канали з використанням HTTPS і REST API. Окремо реалізовано інтеграцію з Grafana для моніторингу ключових метрик і механізми аудиту дій оператора.

Core-компонент є центральним логічним елементом системи та відповідає за обробку даних, аналітику й прийняття керуючих рішень. До його складу входить API Gateway, який забезпечує уніфікований доступ до функціональності системи з використанням токен-орієнтованої автентифікації. Аналіз зібраних даних виконується модулем Pattern Engine на основі поведінкових шаблонів і виявлення аномалій, після чого Scoring Engine здійснює

оцінку рівня ризику для кожного хоста. На основі отриманих оцінок Policy Engine ініціює відповідні сценарії реагування з урахуванням визначених політик безпеки та бази знань загроз.

End Protect Point являє собою компонент захисту кінцевої точки, що розгортається безпосередньо на хості. Основним елементом є агент, який здійснює збір локальних подій і первинний аналіз активності. До його складу входять модулі збору подій операційної системи, аналізу мережевого трафіку на рівні метаданих та локального аналізу поведінки. Для реалізації активних дій реагування використовується адаптер міжмережевого екрана, який взаємодіє з ОС через відповідні API. Обмін даними з Core-компонентом здійснюється через захищені канали з використанням токенів доступу.

Grafana використовується для візуалізації стану системи та відображення ключових показників її роботи. Компонент отримує метрики з Core-компонента через API з доступом у режимі «тільки читання», що зменшує ризики несанкціонованого впливу. Інформаційні панелі дозволяють оператору аналізувати динаміку подій безпеки, навігацію на систему та ефективність застосованих політик.

PostgreSQL виконує роль централізованого сховища даних системи та використовується для збереження облікових даних операторів, токенів агентів, журналів подій і результатів аналізу. Також у базі зберігаються службові структури, зокрема списки блокування. Взаємодія з Core-компонентом здійснюється через захищене TCP-з'єднання, що забезпечує цілісність і надійність зберігання інформації.

Що стосується взаємодії системи, вона представлена як подієво-орієнтований процес між децентралізованими точками моніторингу та центральним ядром аналізу. У процесі роботи агенти фіксують мережеву активність хостів і передають узагальнені метадані до Core-компонента, де відбувається кореляція подій та ідентифікація поведінкових шаблонів. У разі виявлення активності, що відповідає формалізованим ознакам атаки, система виконує оцінку рівня ризику з урахуванням контексту та історії взаємодій.

На основі результатів оцінювання формується керуюче рішення, яке може передбачати обмеження або блокування мережевої взаємодії. Прийняті рішення централізовано зберігаються та використовуються для подальшої кореляції подій іншими компонентами системи. Реалізація реагування здійснюється на рівні операційної системи хоста шляхом автоматизованого застосування правил безпеки.

Таким чином, взаємодія компонентів утворює замкнений цикл, що охоплює збір даних, аналіз, оцінку ризиків та автоматизоване реагування, забезпечуючи своєчасне виявлення та нейтралізацію внутрішніх загроз у локальній мережі.

Література

[1] Stallings W. Network Security Essentials: Applications and Standards. 6th Edition. Pearson. 2020. P. 1–450. ISBN: 978-0134605382.

[2] NIST SP 800-41 Rev. 1. Guidelines on Firewalls and Firewall Policy. National Institute of Standards and Technology. 2009. P. 1–70. DOI: 10.6028/NIST.SP.800-41r1.

[3] Cisco Systems. Designing Secure Network Architectures. Cisco Press. 2017. P. 1–320. ISBN: 978-1587144212.

[4] NIST SP 800-92. Guide to Computer Security Log Management. National Institute of Standards and Technology. 2006. P. 1–36. DOI: 10.6028/NIST.SP.800-92.

[5] MITRE ATT&CK® Framework. MITRE Corporation. 2024. URL: https://attack.mitre.org

[6] Sommer R., Paxson V. Outside the Closed World: On Using Machine Learning for Network Intrusion Detection. Proc. of the IEEE Symposium on Security and Privacy. 2010. P. 305–316. DOI: 10.1109/SP.2010.33.

[7] Behl A., Behl K. Cybertwar: The Next Threat to National Security. Springer. 2016. P. 1–180. ISBN: 978-3319404217.

[8] Cisco NetFlow White Papers. Cisco Systems. 2023. URL: https://www.cisco.com/c/en/us/products/ios-nx-os-software/ios-netflow/index.html

А.3 Конференція «DevOps fwdays’26»