

Міністерство освіти і науки України
Державний університет «Житомирська політехніка»
Факультет інформаційно-комп'ютерних технологій
Кафедра комп'ютерної інженерії та кібербезпеки

Завідувач

кафедри комп'ютерної
інженерії та кібербезпеки

Андрій Єфіменко

2 лютого 2026 р.

ПОЯСНЮВАЛЬНА ЗАПИСКА

до випускної кваліфікаційної роботи бакалавра
на тему: «Проект системи виявлення інцидентів та
комплексного моніторингу на базі Prometheus та
Graylog»

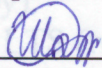
Предмет дослідження – архітектура та методи побудови системи комплексного моніторингу й виявлення інцидентів на основі програмних комплексів Prometheus, Graylog та Snort з підсистемою автоматизованого реагування.

Вихідні дані до роботи:

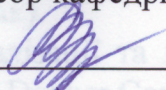
1. Матеріали переддипломної практики
2. Технічна документація програмних засобів Prometheus, Graylog, Snort, Docker
3. Результати аналізу існуючих SIEM-рішень (Splunk, IBM QRadar, Wazuh).

Обладнання та програмне забезпечення:

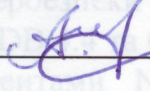
1. Апаратне забезпечення: Виконав: студент 4-го курсу групи КБ-22-2 спеціальності 125 «Кібербезпека»

2. Операційні системи: Ubuntu Server 24.04  Максим ШАТІЛОВ

Керівник: професор кафедри КІ та КБ, д.т.н., доцент

3. Програмне забезпечення віртуалізації:  Володимир ВОРОТНІКОВ

4. Прикладне програмне забезпечення: Рецензент: доцент кафедри КІ та КБ, к.т.н.

5. Платформа Graylog (включаючи MongoDB)  Олексій ШЕЛУХА

тема моніторингу Prometheus (з архітектурою Node Exporter, Nginx Prometheus Exporter, Alertmanager та системою візуалізації Grafana), система виявлення мережеских вторгнень Snort 2.9.x.

5. Засоби розробки: середовище виконання Python 3.12 із бібліотеками (python-telegram-bot, gunicorn).

Житомир – 2026

Державний університет «Житомирська політехніка»
Факультет інформаційно-комп'ютерних технологій
Кафедра комп'ютерної інженерії та кібербезпеки
Спеціальність 125 «Кібербезпека»
Освітня програма «Кібербезпека»

ЗАТВЕРДЖУЮ

Завідувач

кафедри комп'ютерної

інженерії та кібербезпеки

Андрій Єфіменко

2 лютого 2026 р.

ЗАВДАННЯ

на випускню кваліфікаційну роботу бакалавра

Студент: Шатілов Максим Володимирович

Тема роботи: «Проект системи виявлення інцидентів та комплексного моніторингу на базі Prometheus та Graylog»

затверджена Наказом університету від 2 лютого 2026 р. №58/с

Термін здачі студентом закінченої роботи 24 червня 2026 р.

Вихідні дані роботи (зазначається предмет і об'єкт дослідження):

Об'єкт дослідження – процеси моніторингу стану інформаційно-телекомунікаційної інфраструктури та виявлення інцидентів інформаційної безпеки.

Предмет дослідження – архітектура та методи побудови системи комплексного моніторингу й виявлення інцидентів на основі програмних комплексів Prometheus, Graylog та Snort з підсистемою автоматизованого реагування.

Вихідні дані до роботи:

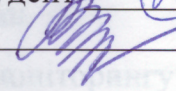
1. Матеріали переддипломної практики
2. Технічна документація програмних засобів Prometheus, Graylog, Snort, Docker
3. Результати аналізу існуючих SIEM-рішень (Splunk, IBM QRadar, Wazuh).

Обладнання та програмне забезпечення:

1. Апаратне забезпечення: персональний комп'ютер для розгортання ізолюваного тестового стенда (вимоги: процесор x86-64, від 4 ГБ оперативної пам'яті, від 30 ГБ дискового простору).
2. Операційні системи: Ubuntu Server 24.04 LTS (гостьова ОС для розгортання цільового вузла та SIEM-сервера), Kali Linux 2026.1 (гостьова ОС для моделювання мережеских атак).
3. Програмне забезпечення віртуалізації та контейнеризації: Oracle VM VirtualBox 7.0.10, Docker 29.5.3, Docker Compose v5.1.4.
4. Прикладне програмне забезпечення кібербезпеки та моніторингу: SIEM-платформа Graylog (включаючи MongoDB 4.4 та OpenSearch 1.3.7), система моніторингу Prometheus (з агентами Node Exporter, Nginx Prometheus Exporter, Alertmanager та системою візуалізації Grafana), система виявлення мережеских вторгнень Snort 2.9.x.
5. Засоби розробки: середовище виконання Python 3.12 із бібліотеками (python-telegram-bot, requests).

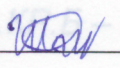
Консультанти з випускної кваліфікаційної роботи із зазначенням розділів, що їх стосуються

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв
1	Фальковський І. Г.	02.02.26	02.02.26 
2	Фальковський І. Г.	02.02.26	02.02.26 
3	Фальковський І. Г.	02.02.26	02.02.26 

Студент  Максим ШАТІЛОВ
Керівник  Володимир ВОРОТНІКОВ

Календарний план

№ з/п	Назви етапів випускної кваліфікаційної роботи	Термін виконання етапів роботи	Примітка
1	Отримання завдання, аналіз технічного завдання, огляд літературних джерел, аналіз аналогів	02.02.2026 – 23.02.2026	Виконано
2	Обґрунтування вибору інструментальних засобів та апаратних вимог	24.02.2026 – 02.03.2026	Виконано
3	Розгортання архітектури тестового стенда та мережевої топології	03.03.2026 – 16.03.2026	Виконано
4	Налаштування підсистеми моніторингу метрик (Prometheus, Grafana)	17.03.2026 – 23.03.2026	Виконано
5	Налаштування SIEM-платформи (Graylog: Streams, Pipelines)	24.03.2026 – 06.04.2026	Виконано
6	Реалізація підсистеми виявлення мережевих вторгнень (Snort)	07.04.2026 – 13.04.2026	Виконано
7	Розробка програмного модуля автоматизованого реагування	14.04.2026 – 27.04.2026	Виконано
8	Тестування програмного комплексу, усунення недоліків	28.04.2026 – 11.05.2026	Виконано
9	Оформлення пояснювальної записки кваліфікаційної роботи	12.05.2026 – 04.06.2026	Виконано
10	Підготовка до захисту, рецензування, нормоконтроль	05.06.2026 – 23.06.2026	Виконано

Студент  Максим ШАТІЛОВ

Керівник  Володимир ВОРОТНІКОВ

Проведено експериментальні АНОТАЦІЯ рацездатності комплексу для семи

Кваліфікаційна робота: 85 сторінки, 20 рисунків, 13 таблиць, 4 додатки, 23 використаних джерел.

Мета роботи – розробка та практична реалізація системи виявлення інцидентів та комплексного моніторингу інформаційної інфраструктури підприємства на основі програмних комплексів Prometheus, Graylog та Snort з функцією автоматизованого реагування на виявлені загрози.

Об'єкт дослідження – процеси моніторингу стану інформаційно-телекомунікаційної інфраструктури та виявлення інцидентів інформаційної безпеки. Предмет дослідження – архітектура та методи побудови системи комплексного моніторингу й виявлення інцидентів на основі програмних комплексів Prometheus, Graylog та Snort з підсистемою автоматизованого реагування.

У роботі застосовано методи системного та порівняльного аналізу для обґрунтування вибору програмних засобів побудови системи, а також методи комп'ютерного моделювання та натурного експерименту для перевірки коректності функціонування розробленого програмного комплексу шляхом імітації мережових атак у контрольованому ізольованому середовищі.

У результаті виконання роботи розгорнуто тестовий стенд із трисегментною мережевою топологією, що відображає типову структуру інфраструктури підприємства та забезпечує безпечне відтворюване моделювання кібератак. Реалізовано підсистему моніторингу інфраструктурних метрик на основі Prometheus, підсистему централізованого аналізу подій безпеки на основі Graylog із застосуванням механізмів потоків та конвеєрів обробки повідомлень, підсистему виявлення мережових вторгнень на основі Snort з набором користувацьких сигнатурних правил, а також програмний модуль автоматизованого реагування, що поєднує аналіз порогових значень спрацювання, віддалене блокування мережових адрес засобами міжмережевого екрана цільового вузла та інтерактивну взаємодію з адміністратором безпеки через месенджер Telegram.

					ККІКБ.КР.Б – 125 – 12 – ПЗ			
Змк.	Арк.	№ докум.	Підпис	Дата	Проект системи виявлення інцидентів та комплексного моніторингу на базі Prometheus та Graylog	Лім.	Арк.	Аркушів
Розроб.		Шатілов М. В.		25.06.26				
Керівник		Воротніков В. В.		25.06.26			4	85
Реценз.		Шелуха О. О.		26.06.26		ФІКТ, Гр. КБ-22-2		
Н. контр.		Воротніков В. В.		25.06.26				
Затверд.		Єфіменко А. А.		26.06.26				

Проведено експериментальну перевірку працездатності комплексу для семи основних категорій подій безпеки, що підтвердила відповідність фактичної поведінки системи обґрунтованій архітектурі.

Практичне значення отриманих результатів полягає у створенні працездатного програмно-апаратного комплексу, який може бути впроваджений в інформаційну інфраструктуру підприємства малого та середнього бізнесу для підвищення рівня захищеності від типових кіберзагроз.

КЛЮЧОВІ СЛОВА: GRAYLOG, КІБЕРБЕЗПЕКА, МОНІТОРИНГ, PROMETHEUS, SNORT.

		Шатілов М. В.			ККІКБ.КР.Б – 125 – 12 – ПЗ	Арк.
		Воротніков В. В.				5
Змн.	Арк.	№ докум.	Підпис	Дата		

ANNOTATION

Qualification thesis: 85 pages, 20 figures, 13 tables, 4 appendices, 23 references.

The purpose of the work is to develop and practically implement an incident detection and comprehensive monitoring system for an enterprise information infrastructure based on the Prometheus, Graylog, and Snort software systems, with an automated threat response capability.

The object of research is the process of monitoring the state of information and telecommunication infrastructure and detecting information security incidents. The subject of research is the architecture and methods of building a comprehensive monitoring and incident detection system based on Prometheus, Graylog, and Snort, supplemented with an automated response subsystem.

The work employs methods of systems analysis and comparative analysis to justify the choice of software tools, as well as computer modelling and physical experimentation to verify the correctness of the developed software system by simulating network attacks in a controlled, isolated environment.

As a result of the work, a test bed with a three-segment network topology was deployed, reflecting the typical structure of an enterprise infrastructure and enabling safe, reproducible simulation of cyberattacks. An infrastructure metrics monitoring subsystem based on Prometheus was implemented, along with a centralised security event analysis subsystem based on Graylog using stream and pipeline processing mechanisms, a network intrusion detection subsystem based on Snort with a set of custom signature rules, and an automated response software module combining threshold analysis, remote firewall-based blocking of the target host, and interactive communication with the security administrator via the Telegram messenger. Experimental verification of the system's functionality was carried out for seven main categories of security events, confirming that the actual system behaviour matches the substantiated architecture.

The practical significance of the results lies in the creation of a functional hardware-software system that can be deployed in the information infrastructure of a small or medium-sized enterprise to improve protection against typical cyber threats.

KEYWORDS: CYBERSECURITY, GRAYLOG, MONITORING, PROMETHEUS, SNORT.

		Шатілов М. В.			ККІКБ.КР.Б – 125 – 12 – ПЗ	Арк.
		Воротніков В. В.				6
Змн.	Арк.	№ докум.	Підпис	Дата		

ЗМІСТ

ВСТУП	9
РОЗДІЛ 1 АНАЛІЗ СУЧАСНИХ ПІДХОДІВ ДО ПОБУДОВИ СИСТЕМ ВИЯВЛЕННЯ ІНЦИДЕНТІВ ТА КОМПЛЕКСНОГО МОНІТОРИНГУ	12
1.1 Аналіз сучасного стану та обґрунтування актуальності теми роботи.....	12
1.2 Існуючі аналоги підсистем кібербезпеки інформаційної системи.....	13
1.3 Архітектура та укрупнена структура програмного комплексу	15
1.4 Інструментальні засоби реалізації програмного продукту та вимоги до апаратних засобів	16
Висновки до першого розділу.....	18
РОЗДІЛ 2 РОЗРОБКА ТА ПРОГРАМНА РЕАЛІЗАЦІЯ СИСТЕМИ ВИЯВЛЕННЯ ІНЦИДЕНТІВ ТА КОМПЛЕКСНОГО МОНІТОРИНГУ	20
2.1 Варіанти використання системи.....	20
2.2 Архітектура системи та мережева топологія тестового стенда	22
2.3 Структура модулів підсистеми моніторингу інфраструктурних метрик та аналізу подій безпеки.....	27
2.4 Реалізація підсистеми виявлення мережевих вторгнень	36
2.5. Компонентна будова та реалізація програмного модуля автоматизованого реагування.....	41
Висновки до другого розділу	51
РОЗДІЛ 3 ОПИС ІНТЕРФЕЙСУ, ПОРЯДОК НАЛАШТУВАННЯ ТА ТЕСТУВАННЯ ПРОГРАМНОГО КОМПЛЕКСУ.....	53
3.1 Порядок встановлення та налаштування програмного комплексу.....	53
3.2 Структура та основні елементи інтерфейсу програмного комплексу	56
3.3 Порядок роботи системи та тестовий приклад	59
3.4 Перевірка коректності функціонування та результати тестування	64
Висновки до третього розділу.....	66
ВИСНОВКИ.....	68
СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ	70
ДОДАТКИ.....	72

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

API – Application Programming Interface, програмний інтерфейс для взаємодії застосунків

DDoS – Distributed Denial of Service, розподілена атака на відмову в обслуговуванні

HTTP – Hypertext Transfer Protocol, протокол передавання гіпертексту

ICMP – Internet Control Message Protocol, протокол керівних повідомлень мережі Інтернет

IDS – Intrusion Detection System, система виявлення мережеских вторгнень

IPS – Intrusion Prevention System, система запобігання мережеским вторгненням

MTTR – Mean Time to Respond, середній час реагування на інцидент

NAT – Network Address Translation, технологія трансляції мережеских адрес

PromQL – Prometheus Query Language, мова запитів системи Prometheus

REST – Representational State Transfer, архітектурний стиль побудови програмних інтерфейсів

SIEM – Security Information and Event Management, система управління інформаційною безпекою та подіями безпеки

SOAR – Security Orchestration, Automation and Response, технологія оркестрації, автоматизації та реагування на інциденти безпеки

SOC – Security Operations Center, центр моніторингу безпеки

SQL – Structured Query Language, мова структурованих запитів до баз даних

SSH – Secure Shell, протокол захищеного віддаленого доступу до вузла

TCP – Transmission Control Protocol, протокол керування передаванням даних

TSDB – Time Series Database, база даних часових рядів

URI – Uniform Resource Identifier, уніфікований ідентифікатор ресурсу

XDR – Extended Detection and Response, технологія розширеного виявлення загроз і реагування

YAML – YAML Ain't Markup Language, формат серіалізації даних, що використовується для конфігураційних файлів

		Шатілов М. В.			ККІКБ.КР.Б – 125 – 12 – ПЗ	Арк.
		Воротніков В. В.				8
Змн.	Арк.	№ докум.	Підпис	Дата		

ВСТУП

Актуальність теми. У сучасних умовах стрімкого розвитку інформаційних технологій та зростання кількості й складності кіберзагроз забезпечення безперервного моніторингу стану інформаційної інфраструктури підприємств набуває критичного значення [5, 6]. Традиційні засоби захисту, зокрема міжмережеві екрани та антивірусне програмне забезпечення, не здатні самотійно забезпечити своєчасне виявлення цілеспрямованих атак, спроб несанкціонованого доступу та аномальної поведінки користувачів у мережі. Для розв'язання цієї проблеми в галузі кібербезпеки використовуються комплексні системи класу SIEM у поєднанні із засобами моніторингу інфраструктурних метрик та системами виявлення мережевих вторгнень (IDS) [1, 2].

Окремої уваги потребує питання автоматизації реагування на виявлені інциденти безпеки, оскільки зростання кількості подій, що генеруються сучасними системами моніторингу, унеможлиблює їх оперативну ручну обробку фахівцем із кібербезпеки. Впровадження механізмів автоматичного та напівавтоматичного реагування, зокрема через інтеграцію з месенджерами для оперативного сповіщення адміністратора, дозволяє суттєво скоротити час між виявленням інциденту та вжиттям заходів щодо його нейтралізації.

Використання рішень з відкритим вихідним кодом, таких як Prometheus, Graylog та Snort, дозволяє побудувати гнучку, масштабовану та економічно ефективну архітектуру системи виявлення інцидентів без значних витрат на ліцензування комерційних аналогів, що є особливо актуальним для підприємств малого та середнього бізнесу. Саме тому дослідження архітектурних рішень та практична реалізація такої системи становить актуальну задачу в межах спеціальності 125 – Кібербезпека.

Мета роботи – розробка та практична реалізація системи виявлення інцидентів та комплексного моніторингу інформаційної інфраструктури підприємства на основі програмних комплексів Prometheus, Graylog та Snort з функцією автоматизованого реагування на виявлені загрози.

Для досягнення поставленої мети у роботі вирішуються такі завдання:

		Шатілов М. В.			ККІКБ.КР.Б – 125 – 12 – ПЗ	Арк.
		Воротніков В. В.				
Змн.	Арк.	№ докум.	Підпис	Дата		9

- проаналізувати сучасний стан технологій моніторингу та виявлення інцидентів інформаційної безпеки, а також існуючі програмні рішення відповідного класу;
- обґрунтувати вибір програмних засобів для побудови системи на основі порівняльного аналізу їх функціональних можливостей;
- розробити архітектуру тестового стенда із розмежуванням мережевих сегментів відповідно до призначення вузлів інфраструктури;
- розгорнути та налаштувати підсистему збору й аналізу метрик апаратного та програмного забезпечення на основі Prometheus;
- налаштувати SIEM-платформу Graylog для централізованого збору, класифікації та кореляції подій безпеки;
- розгорнути систему виявлення мережевих вторгнень Snort з набором правил для детектування основних векторів атак на веб-застосунки;
- розробити програмний модуль автоматизованого реагування на інциденти з інтеграцією у месенджер Telegram та функцією автоматичного блокування шкідливої мережевої активності;
- провести експериментальну перевірку працездатності розробленої системи шляхом моделювання типових кіберінцидентів.

Об’єкт дослідження – процеси моніторингу стану інформаційно-телекомунікаційної інфраструктури та виявлення інцидентів інформаційної безпеки.

Предмет дослідження – архітектура та методи побудови системи комплексного моніторингу й виявлення інцидентів на основі програмних комплексів Prometheus, Graylog та Snort з підсистемою автоматизованого реагування.

Методи дослідження. У роботі застосовано методи системного аналізу для обґрунтування вибору програмних засобів, методи порівняльного аналізу під час оцінки існуючих аналогів систем кібербезпеки, а також методи комп’ютерного моделювання та натурного експерименту для перевірки коректності функціонування розробленого програмного комплексу шляхом імітації мережевих атак у контрольованому ізольованому середовищі відповідно до загальноприйнятої методології тестування на проникнення [23].

		Шатілов М. В.			ККІКБ.КР.Б – 125 – 12 – ПЗ	Арк.
		Воротніков В. В.				
Змн.	Арк.	№ докум.	Підпис	Дата		10

Практичне значення отриманих результатів полягає у створенні працездатного програмно-апаратного комплексу, який може бути впроваджений в інформаційну інфраструктуру підприємства для підвищення рівня захищеності від типових кіберзагроз. Розроблені рекомендації щодо архітектури системи та налаштування правил виявлення інцидентів можуть використовуватися як основа для побудови подібних систем на підприємствах малого та середнього бізнесу.

Структура та обсяг роботи. Кваліфікаційна робота складається зі вступу, трьох розділів, висновків, списку використаних джерел та додатків.

		Шатілов М. В.			ККІКБ.КР.Б – 125 – 12 – ПЗ	Арк.
		Воротніков В. В.				
Змн.	Арк.	№ докум.	Підпис	Дата		11

РОЗДІЛ 1 АНАЛІЗ СУЧАСНИХ ПІДХОДІВ ДО ПОБУДОВИ СИСТЕМ ВИЯВЛЕННЯ ІНЦИДЕНТІВ ТА КОМПЛЕКСНОГО МОНІТОРИНГУ

1.1 Аналіз сучасного стану та обґрунтування актуальності теми роботи

Забезпечення інформаційної безпеки сучасного підприємства неможливе без застосування засобів централізованого моніторингу інфраструктури та аналізу подій безпеки в режимі реального часу. Згідно з усталеною в галузі практикою, повноцінна система виявлення інцидентів повинна охоплювати щонайменше три рівні контролю: моніторинг апаратних і програмних метрик вузлів інфраструктури, централізований збір та кореляцію журналів подій операційних систем і прикладного програмного забезпечення, а також аналіз мережевого трафіку на предмет ознак шкідливої активності.

Кожен із зазначених рівнів реалізується окремим класом спеціалізованого програмного забезпечення. Моніторинг метрик інфраструктури здійснюється системами класу TSDB, до яких належить Prometheus [7]. Централізований збір та аналіз журналів подій є завданням SIEM-платформ, серед яких Graylog [8], Splunk [9] та IBM QRadar [10]. Виявлення мережевих атак забезпечується системами класу IDS/IPS, найпоширенішими представниками яких є Snort [11, 12] та Suricata [13].

Окремою тенденцією останніх років є перехід від суто детективного підходу, що обмежується фіксацією та сповіщенням про інцидент, до автоматизованого реагування (SOAR), коли система самостійно виконує попередньо визначені дії у відповідь на підтверджену загрозу – зокрема блокування мережевої адреси злоумисника. Впровадження елементів такого підходу дозволяє суттєво скоротити середній час реагування на інцидент (MTTR), що є одним із ключових показників ефективності центру моніторингу безпеки (SOC) [3, 4].

Зважаючи на викладене, актуальність теми роботи зумовлена потребою практичної реалізації комплексної системи, яка поєднує усі три рівні моніторингу із підсистемою автоматизованого реагування, на основі рішень з відкритим вихідним кодом, що робить результати дослідження придатними до впровадження на підприємствах із обмеженим бюджетом на засоби кібербезпеки.

		Шатілов М. В.			ККІКБ.КР.Б – 125 – 12 – ПЗ	Арк.
		Воротніков В. В.				
Змн.	Арк.	№ докум.	Підпис	Дата		12

1.2 Існуючі аналоги підсистем кібербезпеки інформаційної системи

Ринок SIEM-рішень представлений як комерційними платформами, так і продуктами з відкритим вихідним кодом. До найпоширеніших комерційних рішень належать Splunk Enterprise Security та IBM QRadar SIEM. Обидві платформи забезпечують централізовану кореляцію подій із сотень джерел, застосування алгоритмів машинного навчання для пріоритизації сповіщень та готові інтеграції з іншими засобами захисту, проте характеризуються високою вартістю ліцензування, що обмежує їх застосування малими та середніми підприємствами.

Серед рішень з відкритим вихідним кодом найбільшого поширення набули Graylog та Wazuh. Wazuh є гібридною платформою класу SIEM/XDR, що використовує агентську архітектуру для моніторингу кінцевих точок, контролю цілісності файлів та надає готові шаблони відповідності нормативним вимогам (PCI DSS, HIPAA, GDPR) [14]. Graylog, на відміну від Wazuh, не вимагає встановлення агентів на кінцевих вузлах і орієнтований на централізований прийом журналів подій за стандартними протоколами (Syslog, GELF), що спрощує інтеграцію з гетерогенною інфраструктурою та є більш доцільним для побудови централізованого вузла збору подій у межах поставленого завдання. Окрім цього, Graylog використовує механізм потоків (Streams) та конвеєрів обробки повідомлень (Pipelines), що дозволяє виконувати класифікацію подій безпеки на етапі їх надходження без необхідності розгортання додаткових компонентів аналізу.

Порівняльна характеристика розглянутих рішень наведена у таблиці 1.1.

Таблиця 1.1

Порівняльна характеристика SIEM-платформ

Критерій	Splunk Enterprise Security	IBM QRadar SIEM	Wazuh	Graylog
Тип ліцензування	Комерційне	Комерційне	Відкритий код / комерційна підтримка	Відкритий код / комерційна підтримка
Архітектура збору даних	Універсальна, без агентів	Універсальна, без агентів	Агентська (на кінцевих точках)	Без агентів, прийом за Syslog/GELF

Критерій	Splunk Enterprise Security	IBM QRadar SIEM	Wazuh	Graylog
Орієнтовна вартість впровадження	Висока	Висока	Низька	Низька
Готові шаблони відповідності нормативам	Наявні	Наявні	Наявні (PCI DSS, HIPAA, GDPR)	Відсутні, потребують власної розробки
Складність початкового налаштування	Висока	Висока	Висока	Середня
Доцільність для централізованого SOC малого підприємства	Низька (вартість)	Низька (вартість)	Середня	Висока

Для виявлення мережових вторгнень розглянуто дві найпоширеніші системи з відкритим вихідним кодом – Snort та Suricata. Snort є більш давнім рішенням з однопотоковою архітектурою обробки трафіку та широкою базою готових правил і документації, що накопичувалися протягом понад двадцяти років використання. Suricata використовує багатопотокову архітектуру, що забезпечує вищу пропускну здатність при аналізі трафіку понад 1 Гбіт/с, та підтримує глибоку інспекцію прикладних протоколів (Deep Packet Inspection) незалежно від сигнатурного аналізу [15]. Водночас за швидкістю виявлення сигнатурних атак на трафіку низької та середньої інтенсивності, що є характерним для тестового стенда підприємства малого бізнесу, обидві системи демонструють порівнянну ефективність. Враховуючи довшу історію застосування, ширшу сумісність із сторонніми інструментами аналізу журналів та простішу процедуру початкового налаштування, для реалізації підсистеми виявлення мережових вторгнень обрано Snort.

1.3 Архітектура та укрупнена структура програмного комплексу

На основі проведеного аналізу визначено укрупнену структуру програмного комплексу, що розробляється в межах роботи. Комплекс реалізує трирівневу архітектуру моніторингу та доповнюється підсистемою автоматизованого реагування на інциденти.

Першим рівнем є підсистема моніторингу інфраструктурних метрик на основі Prometheus, яка опитує спеціалізовані агенти (Node Exporter, Nginx Prometheus Exporter), розгорнуті на цільовому вузлі інфраструктури, та зберігає отримані часові ряди для подальшого аналізу й візуалізації засобами Grafana. Другим рівнем є SIEM-платформа Graylog, що приймає журнали подій операційної системи та системи виявлення вторгнень за протоколом Syslog, виконує їх класифікацію за допомогою механізму потоків і конвеєрів обробки та забезпечує зберігання подій у пошуковому індексі OpenSearch. Третім рівнем є підсистема виявлення мережових вторгнень Snort, яка аналізує мережовий трафік цільового вузла на відповідність набору сигнатур типових атак на веб-застосунки та передає результати аналізу до Graylog.

Завданням архітектури тестового стенда є відображення типової структури інформаційної інфраструктури підприємства з розмежуванням мережових сегментів за функціональним призначенням: сегмент управління, у якому розташовано вузол центру моніторингу безпеки (SIEM/SOC-сервер), та публічний сегмент, що емулює доступність веб-сервісу підприємства для зовнішніх користувачів і, відповідно, для потенційного злоумисника. Така сегментація дозволяє безпечно й відтворювано моделювати мережові атаки із виокремленого вузла без ризику для сегмента управління.

Четвертим, надбудовним компонентом архітектури є підсистема автоматизованого реагування, реалізована у вигляді програмного модуля, що опитує програмний інтерфейс Graylog, аналізує класифіковані події на відповідність попередньо визначеним порогам спрацювання та виконує одну з двох дій: автоматичне блокування мережової адреси джерела атаки засобами міжмережевого екрана цільового вузла або формування сповіщення відповідальній особі через месенджер Telegram

		Шатілов М. В.			ККІКБ.КР.Б – 125 – 12 – ПЗ	Арк.
		Воротніков В. В.				
Змн.	Арк.	№ докум.	Підпис	Дата		15

з можливістю ухвалення рішення про блокування в інтерактивному режимі. Узагальнена структура взаємодії компонентів комплексу наведена на рисунку 1.1.

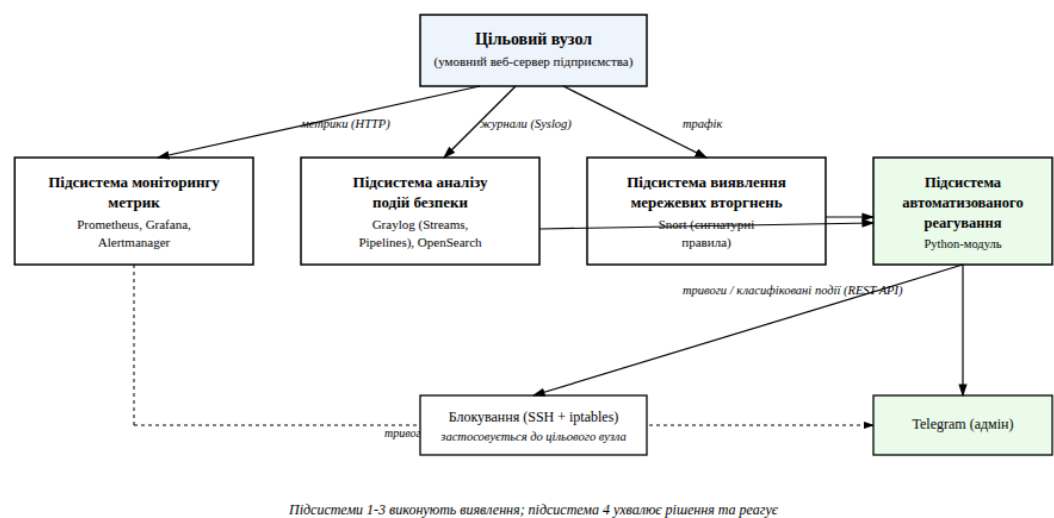


Рисунок 1.1 – Укрупнена структура програмного комплексу

Детальний опис архітектури тестового стенда, конфігурації мережесегментів, структури модулів програмного комплексу та алгоритмів обробки подій наведено у другому розділі роботи.

1.4 Інструментальні засоби реалізації програмного продукту та вимоги до апаратних засобів

Для реалізації програмного комплексу обрано стек технологій з відкритим вихідним кодом, що відповідає вимогам економічної ефективності та гнучкості налаштування, визначеним у підрозділі 1.1. Розгортання компонентів комплексу здійснюється засобами контейнеризації Docker та Docker Compose, що забезпечує ізоляцію сервісів та відтворюваність середовища розгортання [16]. Використання контейнерної архітектури є критично важливим для даного проєкту, оскільки воно дозволяє уникнути конфліктів системних залежностей між різними складовими SIEM-платформи (базами даних, пошуковими індексами та веб-серверами).

Перелік основних програмних засобів, використаних під час реалізації комплексу, наведено у таблиці 1.2.

Перелік програмних засобів реалізації комплексу

Компонент	Призначення	Версія
Prometheus	Збір та зберігання метрик апаратного й програмного забезпечення	latest
Node Exporter	Агент експорту системних метрик операційної системи	1.7.0
Nginx Prometheus Exporter	Агент експорту метрик веб-сервера Nginx	1.1.0
Grafana	Візуалізація метрик, отриманих від Prometheus	latest
Alertmanager	Маршрутизація сповіщень Prometheus до Telegram	latest
MongoDB	Зберігання конфігурації SIEM-платформи Graylog	4.4
OpenSearch	Пошуковий індекс для зберігання журналів подій	1.3.7
Graylog	SIEM-платформа централізованого аналізу подій	4.3
Snort	Система виявлення мережових вторгнень (IDS)	2.9.x
Python 3 / python-telegram-bot	Модуль автоматизованого реагування на інциденти	3.12 / 13.15
Docker / Docker Compose	Контейнеризація компонентів комплексу	29.5.3 / v5.1.4

Апаратні вимоги до тестового стенда визначаються кількістю та призначенням віртуальних машин, що входять до його складу. Стенд реалізовано засобами платформи віртуалізації VirtualBox у складі трьох віртуальних машин: сервера SIEM/SOC, цільового вузла (умовного веб-сервера підприємства) та робочої станції адміністратора безпеки.

Враховуючи ресурсоемність компонентів пошукового індексу OpenSearch та SIEM-платформи Graylog, для сервера SIEM/SOC визначено мінімальні вимоги: не менше 4 ГБ оперативної пам'яті та 30 ГБ дискового простору. Для цільового вузла достатньо 2 ГБ оперативної пам'яті та 20 ГБ дискового простору. Окремою вимогою, виявленою експериментально під час розгортання комплексу, є відсутність

		Шатілов М. В.			ККІКБ.КР.Б – 125 – 12 – ПЗ	Арк.
		Воротніков В. В.				
Змн.	Арк.	№ докум.	Підпис	Дата		17

необхідності підтримки набору інструкцій AVX центральним процесором хост-системи для версії MongoDB 5.0 та вище [17], що зумовило застосування у роботі версій компонентів MongoDB 4.4, Graylog 4.3 та OpenSearch 1.3.7 як сумісних між собою та з типовим апаратним забезпеченням середовища віртуалізації.

Цільовий вузол інфраструктури конфігурується з трьома мережевими адаптерами: адаптером NAT для забезпечення доступу до мережі Інтернет з метою оновлення програмного забезпечення, адаптером внутрішньої мережі управління для взаємодії з сервером SIEM/SOC та адаптером публічного сегмента мережі, призначеним для емуляції доступності веб-сервісу підприємства та моделювання мережових атак із виокремленого вузла зловмисника.

Висновки до першого розділу

У результаті аналізу сучасного стану технологій моніторингу та виявлення інцидентів інформаційної безпеки встановлено, що повноцінна система такого класу повинна охоплювати три взаємодоповнювальні рівні контролю: моніторинг інфраструктурних метрик, централізований аналіз журналів подій засобами SIEM-платформи та виявлення мережових вторгнень засобами IDS. Визначено, що автоматизація реагування на інциденти є самостійним та дедалі важливішим напрямком розвитку систем класу SOC, оскільки дозволяє суттєво скоротити час між виявленням загрози та вжиттям заходів щодо її нейтралізації.

Порівняльний аналіз існуючих програмних рішень показав, що для централізованого збору та аналізу подій безпеки найбільш доцільним у межах поставленого завдання є застосування SIEM-платформи Graylog, яка, на відміну від комерційних аналогів Splunk та IBM QRadar, не потребує витрат на ліцензування, а порівняно з альтернативним рішенням з відкритим вихідним кодом Wazuh не вимагає розгортання агентів на кінцевих вузлах інфраструктури. Для виявлення мережових вторгнень обґрунтовано вибір системи Snort з огляду на простішу процедуру початкового налаштування та достатню ефективність сигнатурного аналізу за

		Шатілов М. В.			ККІКБ.КР.Б – 125 – 12 – ПЗ	Арк.
		Воротніков В. В.				18
Змн.	Арк.	№ докум.	Підпис	Дата		

умов трафіку низької та середньої інтенсивності, характерного для тестового стенда підприємства малого бізнесу.

Визначено укрупнену структуру програмного комплексу, що складається з чотирьох взаємопов'язаних підсистем: моніторингу інфраструктурних метрик на основі Prometheus, централізованого аналізу подій на основі Graylog, виявлення мережних вторгнень на основі Snort та автоматизованого реагування на інциденти, реалізованої програмним модулем з інтеграцією у месенджер Telegram. Обґрунтовано архітектуру тестового стенда із розмежуванням мережних сегментів управління та публічного доступу, що забезпечує безпечне відтворюване моделювання кібератак.

Визначено перелік інструментальних засобів реалізації програмного комплексу та вимоги до апаратного забезпечення тестового стенда, зокрема встановлено обмеження щодо сумісності окремих версій компонентів SIEM-платформи з набором інструкцій центрального процесора середовища віртуалізації, що враховано під час вибору конкретних версій програмного забезпечення. Результати проведеного аналізу становлять основу для розроблення архітектури та програмної реалізації комплексу, опис яких наведено у другому розділі роботи.

		Шатілов М. В.			ККІКБ.КР.Б – 125 – 12 – ПЗ	Арк.
		Воротніков В. В.				
Змн.	Арк.	№ докум.	Підпис	Дата		19

РОЗДІЛ 2 РОЗРОБКА ТА ПРОГРАМНА РЕАЛІЗАЦІЯ СИСТЕМИ ВИЯВ- ЛЕННЯ ІНЦИДЕНТІВ ТА КОМПЛЕКСНОГО МОНІТОРИНГУ

2.1 Варіанти використання системи

Визначення варіантів використання (Use Cases) розробленого програмного комплексу дозволяє формалізувати функціональні вимоги до системи та окреслити сценарії взаємодії її дійових осіб з окремими підсистемами. У межах роботи виокремлено двох дійових осіб: адміністратора безпеки, який здійснює моніторинг стану інфраструктури та ухвалює рішення щодо реагування на інциденти, та зловмисника, дії якого ініціюють спрацювання механізмів виявлення.

Основними варіантами використання з боку адміністратора безпеки є: перегляд поточного стану інфраструктурних метрик цільового вузла засобами Grafana; перегляд журналів подій та класифікованих інцидентів засобами Graylog; отримання сповіщення про виявлений інцидент через месенджер Telegram; ухвалення рішення про блокування або ігнорування підозрілої мережевої адреси шляхом натискання інтерактивної кнопки в повідомленні Telegram; перегляд переліку заблокованих адрес та ручне розблокування конкретної адреси за допомогою команд бота.

З боку зловмисника варіантами використання, що ініціюють реакцію системи, є: спроба підбору облікових даних протоколу SSH; спроба підвищення привілеїв через виконання команди sudo; спроба безпосереднього доступу до облікового запису суперкористувача; спроба впровадження SQL-коду через параметри HTTP-запиту; сканування мережевих портів цільового вузла; сканування каталогів та файлів веб-застосунку; використання автоматизованих засобів сканування вразливостей. Детальне визначення цих векторів атак є необхідним кроком для точного налаштування сигнатур виявлення у підсистемі Snort та конвеєрів обробки платформи Graylog. Узагальнена діаграма варіантів використання, що наочно відображає взаємодію обох дійових осіб із розробленим програмним комплексом, наведена на рисунку 2.1.»

		Шатілов М. В.			ККІКБ.КР.Б – 125 – 12 – ПЗ	Арк.
		Воротніков В. В.				
Змн.	Арк.	№ докум.	Підпис	Дата		20

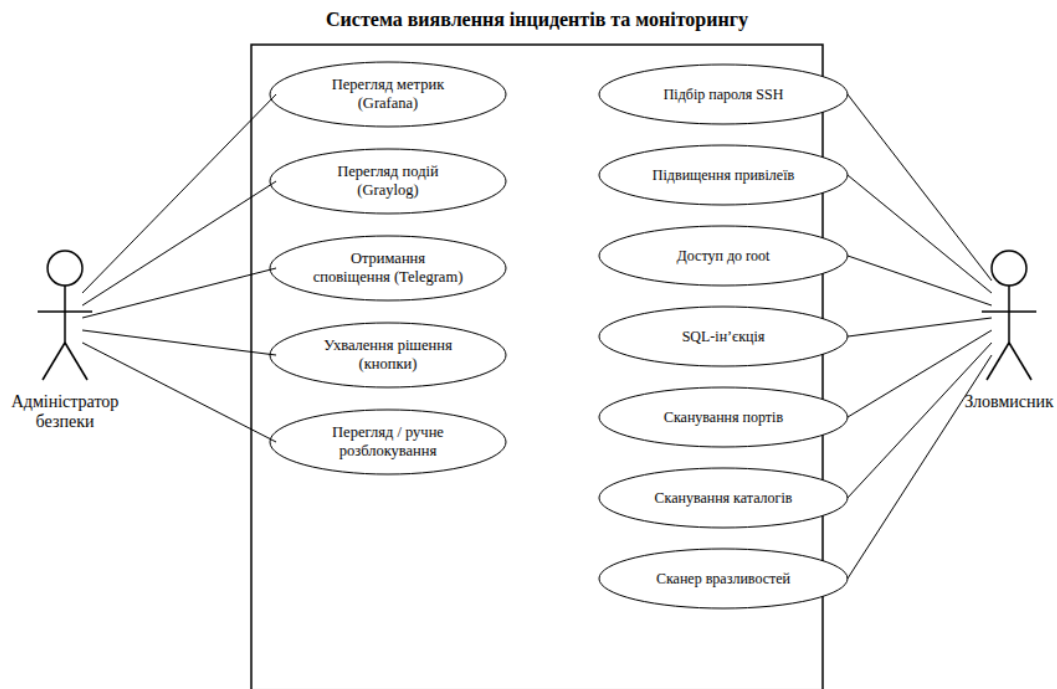


Рисунок 2.1 – Діаграма варіантів використання системи

Для трьох найбільш характерних варіантів використання, що визначають основну функціональність підсистеми автоматизованого реагування, наведено детальну специфікацію у вигляді опису передумови, основного сценарію та результату виконання (таблиця 2.1).

Таблиця 2.1

Детальна специфікація характерних варіантів використання системи

Елемент специфікації	Сканування мережевих портів	Підбір пароля SSH	Ухвалення рішення адміністратором
Дійова особа	Зловмисник	Зловмисник	Адміністратор безпеки
Передумова	Цільовий вузол доступний у сегменті vlan-public	Служба SSH запущена на цільовому вузлі	Подія безпеки не досягла порогового значення автоблокування
Основний сценарій	Зловмисник надсилає послідовність TCP-пакетів з прапором SYN на різні порти цільового вузла	Зловмисник виконує послідовні спроби автентифікації з неправильним паролем	Адміністратор отримує сповіщення з кнопками та обирає дію

Елемент специфікації	Сканування мережевих портів	Підбір пароля SSH	Ухвалення рішення адміністратором
Реакція системи	Snort реєструє перевищення порогового значення (5 пакетів за 5 секунд) та формує подію Port Scan	Журнал auth.log фіксує записи Failed password, які класифікуються потоком SSH Brute-force	Модуль виконує блокування або позначає подію як проігноровану
Результат	Мережева адреса джерела автоматично блокується (поріг – 1 подія)	Мережева адреса блокується після п'ятої невдалої спроби за 60 секунд	Подія вилючається зі списку очікування рішення

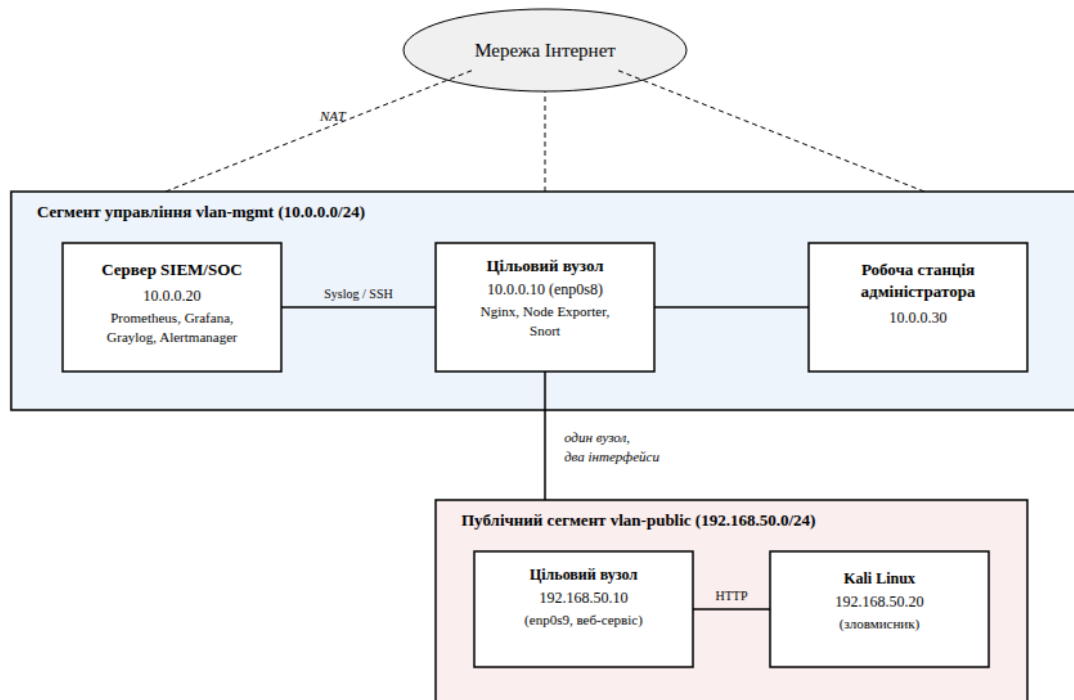
Наведена специфікація ілюструє принципову відмінність між варіантами використання, що завершуються автоматичним блокуванням без участі людини (сканування портів, де поріг спрацювання дорівнює одній події з огляду на вбудований механізм порогового аналізу самого Snort), та варіантами, що потребують накопичення кількох подій у часовому вікні (підбір пароля SSH, де поріг автоблокування становить п'ять подій за 60 секунд). Обґрунтування конкретних числових значень порогів для кожної категорії подій наведено у третьому розділі роботи за результатами експериментальної перевірки.

Кожен із зазначених варіантів використання реалізується послідовністю дій відповідних підсистем програмного комплексу, детальний опис структури та взаємодії яких наведено у наступних підрозділах.

2.2 Архітектура системи та мережева топологія тестового стенда

Тестовий стенд реалізовано засобами платформи віртуалізації VirtualBox у складі трьох віртуальних машин, кожна з яких виконує самостійну функціональну роль: сервера SIEM/SOC, цільового вузла (умовного веб-сервера підприємства) та робочої станції адміністратора безпеки. Розмежування функціональних ролей між вузлами відповідає типовій структурі центру моніторингу безпеки, де сервер аналізу подій фізично відокремлений від об'єкта моніторингу.

Мережева топологія стенда побудована на основі трьох ізольованих сегментів. Перший сегмент реалізує мережу управління (vlan-mgmt) з адресним простором 10.0.0.0/24, до якого підключені всі три вузли стенда: сервер SIEM/SOC з адресою 10.0.0.20, цільовий вузол з адресою 10.0.0.10 та робоча станція адміністратора з адресою 10.0.0.30. Цей сегмент забезпечує передавання журналів подій від цільового вузла до сервера SIEM/SOC за протоколом Syslog та віддалене виконання команд блокування засобами протоколу SSH. Другий сегмент – публічний сегмент мережі (vlan-public) з адресним простором 192.168.50.0/24, призначений виключно для емуляції доступності веб-сервісу підприємства для зовнішніх користувачів; до нього підключено інтерфейс цільового вузла з адресою 192.168.50.10 та віртуальну машину Kali Linux, що виконує роль зовнішнього зловмисника. Третій сегмент реалізовано засобами адаптера NAT і використовується виключно для доступу віртуальних машин до мережі Інтернет із метою встановлення та оновлення програмного забезпечення. Схема мережевої топології тестового стенда наведена на рисунку 2.2.



Примітка: робоча станція адміністратора та сервер SIEM/SOC не мають мережевого маршруту до сегмента vlan-public.

Рисунок 2.2 – Схема мережевої топології тестового стенда

		Шатілов М. В.			ККІКБ.КР.Б – 125 – 12 – ПЗ	Арк.
		Воротніков В. В.				
Змн.	Арк.	№ докум.	Підпис	Дата		23

Детальний перелік мережевих інтерфейсів цільового вузла та їхнє призначення наведено у таблиці 2.2.

Таблиця 2.2

Перелік мережевих інтерфейсів цільового вузла

Інтерфейс	Мережа VirtualBox	ІР-адреса	Призначення
enp0s3	NAT	динамічна (DHCP)	Доступ до мережі Інтернет для оновлення пакетів операційної системи
enp0s8	Internal Network (vlan-mgmt)	10.0.0.10/24	Передавання журналів подій до Graylog, прийом команд блокування через SSH
enp0s9	Internal Network (vlan-public)	192.168.50.10/24	Обслуговування HTTP-запитів веб-сервісу, аналіз трафіку засобами Snort

Статична конфігурація мережевих інтерфейсів enp0s8 та enp0s9 виконана засобами системи керування мережею Netplan, що є стандартним засобом конфігурації мережі в операційній системі Ubuntu Server.

Фрагмент конфігураційного файлу Netplan для мережевих інтерфейсів цільового вузла:

```
network:
  ethernets:
    enp0s3:
      dhcp4: true
    enp0s8:
      dhcp4: false
      addresses: [10.0.0.10/24]
    enp0s9:
      dhcp4: false
      addresses: [192.168.50.10/24]
  version: 2
```

Застосування декларативного підходу до конфігурації мережі, притаманного Netplan, забезпечує детерміновану відповідність мережевого інтерфейсу та присвоєної йому адреси після кожного перезавантаження віртуальної машини, на

		Шатілов М. В.			ККІКБ.КР.Б – 125 – 12 – ПЗ	Арк.
		Воротніков В. В.				
Змн.	Арк.	№ докум.	Підпис	Дата		24

відміну від конфігурації засобами графічного інтерфейсу керування мережею, що під час експериментального розгортання стенда виявилася ненадійною для застосування до серверних систем без графічного середовища через втрату налаштувань після перезапуску. Аналогічним чином налаштовано мережеві інтерфейси сервера SIEM/SOC та робочої станції адміністратора, кожен з яких підключений лише до сегмента мережі управління (vlan-mgmt) з адресами 10.0.0.20 та 10.0.0.30 відповідно.

Принциповою архітектурною вимогою є відсутність мережевої доступності сегмента vlan-public з боку сервера SIEM/SOC та робочої станції адміністратора, що відтворює реальну ситуацію, коли зовнішній зломисник не має прямого мережевого доступу до внутрішнього сегмента управління підприємства і може взаємодіяти лише з публічно доступним веб-сервісом. Така сегментація також виконує функцію засобу безпеки самого процесу дослідження, оскільки унеможливорює випадковий вплив тестових атак на сервер SIEM/SOC. Перевірка дотримання цієї вимоги виконувалася шляхом спроби встановлення мережевого з'єднання з робочої станції адміністратора до інтерфейсу цільового вузла в сегменті vlan-public, що, відповідно до очікуваної поведінки, завершувалася помилкою через відсутність маршруту до вказаної мережі.

Програмні компоненти сервера SIEM/SOC розгортаються засобами контейнеризації Docker Compose, що дозволяє ізолювати кожен сервіс в окремому контейнері та забезпечити відтворюваність середовища розгортання. Конфігураційний файл docker-compose.yml визначає шість сервісів: mongodb, opensearch, graylog, prometheus, grafana та alertmanager, об'єднаних спільною віртуальною мережею Docker, що забезпечує їхню взаємодію за внутрішніми мережевими іменами без необхідності використання адрес хост-системи.

Фрагмент конфігураційного файлу docker-compose.yml для сервісу Graylog:

```
graylog:
  image: graylog/graylog:4.3
  environment:
    - GRAYLOG_PASSWORD_SECRET=*****
    - GRAYLOG_ROOT_PASSWORD_SHA2=*****
```

		Шатілов М. В.			ККІКБ.КР.Б – 125 – 12 – ПЗ	Арк.
		Воротніков В. В.				
Змн.	Арк.	№ докум.	Підпис	Дата		25

```

- GRAYLOG_HTTP_BIND_ADDRESS=0.0.0.0:9000
- GRAYLOG_MONGODB_URI=mongodb://mongodb:27017/graylog
- GRAYLOG_ELASTICSEARCH_HOSTS=http://opensearch:9200
ports:
- "9000:9000"
- "1514:1514/udp"
volumes:
- graylog_data:/usr/share/graylog/data
depends_on:
- opensearch
- mongodb
restart: unless-stopped

```

Параметри GRAYLOG_MONGODB_URI та GRAYLOG_ELASTICSEARCH_HOSTS визначають адреси залежних сервісів зберігання даних за їхніми іменами в мережі Docker, що унеможливлює необхідність використання статичних IP-адрес контейнерів, які можуть змінюватися під час перестворення. Параметр volumes монтує іменований том graylog_data до каталогу даних Graylog у середині контейнера; застосування іменованого тому, а не анонімного, є визначальним для забезпечення сталості ідентифікатора вузла Graylog (Node ID) між перезапусками контейнера, що підтверджено експериментально під час налагодження комплексу – у разі відсутності цього тому повторне створення контейнера призводило до генерації нового ідентифікатора вузла та, відповідно, до необхідності повторного створення вхідного модуля прийому журналів подій. Параметр restart зі значенням unless-stopped забезпечує автоматичний перезапуск контейнера після перезавантаження хост-системи без необхідності ручного втручання адміністратора.

Аналогічний підхід застосовано до решти сервісів комплексу. Особливістю конфігурації сервісів mongodb та opensearch є фіксація конкретних версій програмного забезпечення (4.4 та 1.3.7 відповідно) замість використання тегу latest, що в інших сервісах є прийнятним, оскільки забезпечує контрольоване відтворення середовища розгортання та враховує виявлене обмеження сумісності із набором інструкцій центрального процесора, описане у підрозділі 1.4.

		Шатілов М. В.			ККІКБ.КР.Б – 125 – 12 – ПЗ	Арк.
		Воротніков В. В.				
Змн.	Арк.	№ докум.	Підпис	Дата		26

Кожен сервіс комплексу публікує визначений мережевий порт на хост-системі для забезпечення доступу до веб-інтерфейсу або програмного інтерфейсу управління. Перелік опублікованих портів та їхнє функціональне призначення наведено у таблиці 2.3.

Таблиця 2.3

Перелік опублікованих мережевих портів сервісів сервера SIEM/SOC

Сервіс	Порт	Призначення
Graylog	9000/tcp	Веб-інтерфейс управління та програмний інтерфейс REST API
Graylog	1514/udp	Прийом журналів подій за протоколом Syslog
Prometheus	9090/tcp	Веб-інтерфейс перегляду метрик та стану правил тривоги
Grafana	3000/tcp	Веб-інтерфейс візуалізації метрик
Alertmanager	9093/tcp	Веб-інтерфейс перегляду активних тривог

Збереження даних кожного сервісу між перезапусками контейнерів забезпечується застосуванням іменованих томів Docker (named volumes), визначених у секції volumes конфігураційного файлу: mongodb_data для конфігурації Graylog, os_data для індексованих журналів подій OpenSearch, prometheus_data для часових рядів метрик та grafana_data для налаштувань дашбордів Grafana. Застосування іменованих, а не анонімних томів дозволяє звертатися до збережених даних за стабільним ім'ям незалежно від ідентифікатора контейнера, що формується середовищем Docker під час кожного запуску, та є необхідною умовою коректного функціонування комплексу після перезавантаження хост-системи.

2.3 Структура модулів підсистеми моніторингу інфраструктурних метрик та аналізу подій безпеки

Підсистема моніторингу інфраструктурних метрик реалізована на основі системи Prometheus, що функціонує за pull-моделлю, самостійно опитуючи спеціалізовані агенти експорту метрик, розгорнуті на цільовому вузлі. Агент Node Exporter надає метрики завантаження центрального процесора, використання оперативної пам'яті, дискового простору та мережевого трафіку операційної системи

		Шатілов М. В.			ККІКБ.КР.Б – 125 – 12 – ПЗ	Арк.
		Воротніков В. В.				
Змн.	Арк.	№ докум.	Підпис	Дата		27

[18]. Агент Nginx Prometheus Exporter надає метрики веб-сервера: кількість активних з'єднань, інтенсивність запитів та розподіл кодів стану HTTP-відповідей. Зібрані метрики накопичуються у вбудованій базі часових рядів Prometheus та використовуються підсистемою генерації тривог (Alerts), правила якої визначають критичні порогові значення для кожної метрики.

Структуру та параметри основних правил генерації тривог наведено у таблиці 2.4.

Таблиця 2.4

Правила генерації тривог підсистеми Prometheus

Назва правила	Умова спрацювання	Тривалість	Рівень критичності
HighCPUUsage	Завантаження CPU понад 85%	3 хвилини	Попередження
HighMemoryUsage	Використання RAM понад 90%	3 хвилини	Попередження
DiskSpaceRunningOut	Вільний дисковий простір менше 10%	5 хвилин	Попередження
HighNetworkTraffic	Вхідний трафік понад 10 МБ/с	2 хвилини	Попередження
InstanceDown	Відсутність відповіді від вузла	1 хвилина	Критичний
UnauthorizedReboot	Час роботи вузла менше 5 хвилин	1 хвилина	Критичний

Конфігурація основного файлу prometheus.yml визначає інтервал опитування цільових вузлів та перелік завдань збору метрик (scrape_configs), кожне з яких відповідає окремому джерелу даних.

Фрагмент конфігураційного файлу prometheus.yml:

```
global:
  scrape_interval: 15s
  evaluation_interval: 15s

rule_files:
  - "/etc/prometheus/alerts.yml"

alerting:
  alertmanagers:
```

```

- static_configs:
  - targets: ["alertmanager:9093"]

scrape_configs:
- job_name: "target_server"
  static_configs:
    - targets: ["10.0.0.10:9100"]
- job_name: "target_nginx"
  static_configs:
    - targets: ["10.0.0.10:9113"]

```

Параметр `scrape_interval` визначає періодичність опитування цільових вузлів і встановлений на рівні 15 секунд як компромісне значення між оперативністю виявлення аномалій та навантаженням на мережу й систему зберігання часових рядів. Секція `rule_files` підключає окремий файл `alerts.yml`, що містить логічні правила генерації тривоги, відокремлені від основної конфігурації з метою зручності їх супроводження. Секція `scrape_configs` визначає дві окремі задачі опитування: `target_server` для отримання системних метрик через агент Node Exporter на порту 9100, та `target_nginx` для отримання метрик веб-сервера через агент Nginx Prometheus Exporter на порту 9113, обидва розгорнуті на цільовому вузлі за адресою 10.0.0.10 у сегменті мережі управління.

Структуру правила генерації тривоги в окремому файлі `alerts.yml` розглянемо на прикладі правила `HighCPUUsage`.

Фрагмент конфігураційного файлу `alerts.yml` для правила `HighCPUUsage`:

```

- alert: HighCPUUsage
  expr: 100 - (avg by (instance)
    (rate(node_cpu_seconds_total{mode="idle"}[1m])) * 100) > 85
  for: 3m
  labels:
    severity: warning
  annotations:
    summary: "Високе навантаження CPU"
    description: "CPU завантажено понад 85% протягом 3 хвилин."

```

		Шатілов М. В.			ККІКБ.КР.Б – 125 – 12 – ПЗ	Арк.
		Воротніков В. В.				
Змн.	Арк.	№ докум.	Підпис	Дата		29

Поле `expr` визначає вираз мови запитів PromQL, що обчислює відсоток завантаження процесора як доповнення до частки часу перебування в стані `idle`, усередненої за останню хвилину для кожного контрольованого вузла (за допомогою функції `rate` та агрегувальної функції `avg by`). Поле `for` визначає тривалість, протягом якої умова має залишатися істинною безперервно, перш ніж тривога перейде у стан `firing` – застосування цього поля є визначальним для запобігання формуванню тривоги через короточасні пікові навантаження, що не свідчать про реальну проблему інфраструктури. Поля `labels` та `annotations` визначають метадані тривоги, що використовуються підсистемою Alertmanager під час маршрутизації та формування тексту сповіщення.

Результат функціонування підсистеми моніторингу інфраструктурних метрик у вигляді дашборду Grafana з відображенням реальних значень навантаження на центральний процесор, оперативну пам'ять та час безперервної роботи цільового вузла наведено на рисунку 2.3.

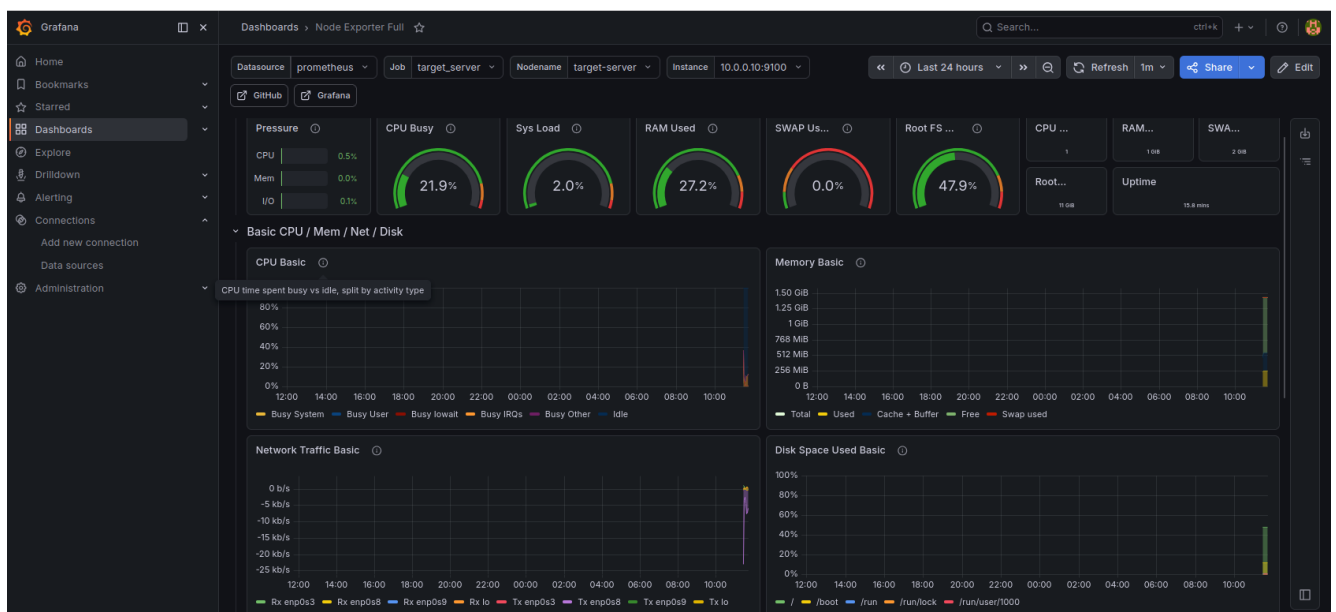


Рисунок 2.3 – Приклад відображення метрик цільового вузла у дашборді Grafana

При спрацюванні правила тривоги передається компоненту Alertmanager, конфігураційний файл якого визначає маршрут доставки сповіщень та формат тексту повідомлення.

Фрагмент конфігураційного файлу `alertmanager.yml`:

```
route:
```

		Шатілов М. В.			ККІКБ.КР.Б – 125 – 12 – ПЗ	Арк.
		Воротніков В. В.				
Змн.	Арк.	№ докум.	Підпис	Дата		30

```

group_by: ['alertname']
group_wait: 10s
group_interval: 10s
repeat_interval: 1h
receiver: 'telegram'

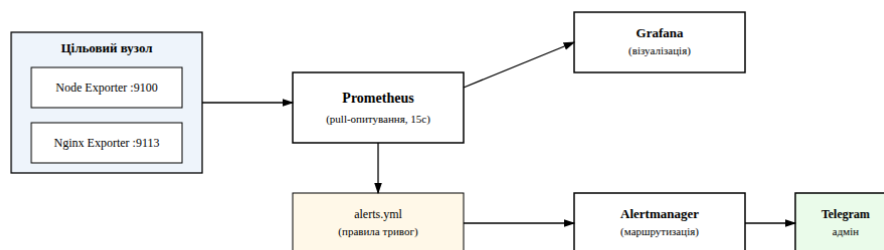
receivers:
- name: 'telegram'
  telegram_configs:
    - bot_token: '*****'
      chat_id: *****
      parse_mode: 'HTML'
      message: '{{ template "telegram.message" . }}'

```

Параметр `group_by` визначає поле, за яким однакові тривоги, що надходять повторно протягом короткого проміжку часу, об'єднуються в одне повідомлення замість надсилання дублікатів. Параметри `group_wait` та `group_interval` встановлюють витримку перед першим та повторним надсиланням сповіщення в межах однієї групи, а параметр `repeat_interval` обмежує частоту повторного надсилання вже відомого активного сповіщення до одного разу на годину з метою запобігання надмірному навантаженню на канал комунікації адміністратора. Текст повідомлення формується за окремим шаблоном, що використовує мову шаблонів Go template та звертається до полів `CommonLabels` та `Annotations`, визначених безпосередньо у правилі тривоги Prometheus, що дозволяє формувати змістовний текст сповіщення без необхідності дублювання інформації в конфігурації Alertmanager. Такий модульний підхід до маршрутизації та шаблонізації забезпечує високу гнучкість системи моніторингу. Завдяки механізмам групування та обмеження частоти повторних тривог, Alertmanager ефективно запобігає інформаційному перевантаженню адміністратора безпеки під час масових збоїв, гарантуючи при цьому оперативну доставку критичних сповіщень.

Узагальнена структура взаємодії компонентів підсистеми моніторингу інфраструктурних метрик – від агентів збору даних на цільовому вузлі до доставки сповіщення засобами Telegram – наведена на рисунку 2.4.

		Шатілов М. В.			ККІКБ.КР.Б – 125 – 12 – ПЗ	Арк.
		Воротніков В. В.				
Змн.	Арк.	№ докум.	Підпис	Дата		31



Параметр PromQL-вираз $100 - (\text{avg}(\text{rate}(\text{node_cpu_seconds_total}[\text{mode}=\text{"idle"}][1m])) * 100) > 85$
обчислюється підсистемою Prometheus кожні 15 секунд для кожного підключеного вузла

Рисунок 2.4 – Структура підсистеми моніторингу інфраструктурних метрик

Підсистема централізованого аналізу подій безпеки реалізована на основі SIEM-платформи Graylog, що використовує пошуковий рушій OpenSearch для зберігання та індексування подій [19]. Журнали подій операційної системи цільового вузла передаються засобом rsyslog за протоколом Syslog на визначений порт сервера SIEM/SOC, де приймаються вхідним модулем Graylog (Input). Конфігурація служби rsyslog на цільовому вузлі доповнюється правилом пересилання усіх повідомлень журналу на визначену адресу та порт сервера SIEM/SOC за протоколом UDP.

Фрагмент конфігураційного файлу пересилання журналів rsyslog:

```
# /etc/rsyslog.d/graylog.conf
*. * @10.0.0.20:1514
```

Структурування вхідного потоку подій виконується за допомогою механізму потоків (Streams), кожен з яких визначається одним або декількома правилами фільтрації за вмістом текстового поля повідомлення. Повний перелік реалізованих потоків та відповідних правил фільтрації наведено у таблиці 2.5.

Таблиця 2.5

Перелік потоків (Streams) та правил фільтрації подій

Назва потоку	Правило фільтрації (умова: message contains)
[SIEM] SSH Brute-force Attempts	Failed password
[SIEM] Privilege Escalation (Sudo Failures)	pam_unix(sudo:auth) TA authentication failure
[SIEM] Direct Root Access Attempts	FAILED SU (to root)

Назва потоку	Правило фільтрації (умова: message contains)
[SIEM] Unauthorized Account Creation	new user: name=
[SIEM] Cron Modifications (Persistence)	crontab
[SIEM] Snort IDS Alerts	[1: (характерний префікс ідентифікатора правила Snort)

Інтерфейс керування переліком реалізованих потоків у веб-консолі Graylog, що відображає назву, статус (Running) та кількість правил кожного потоку, наведено на рисунках 2.5 та 2.6.

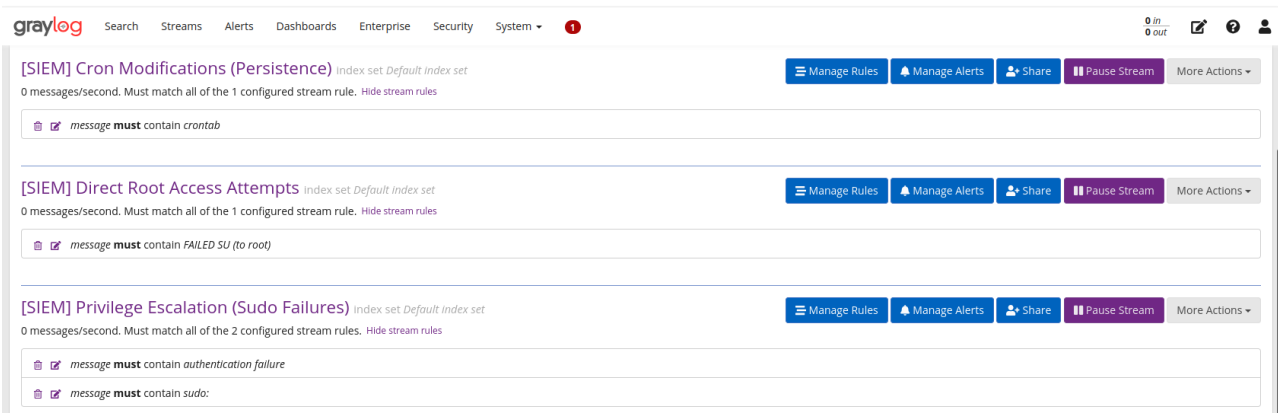


Рисунок 2.5 – Інтерфейс керування потоками у веб-консолі Graylog

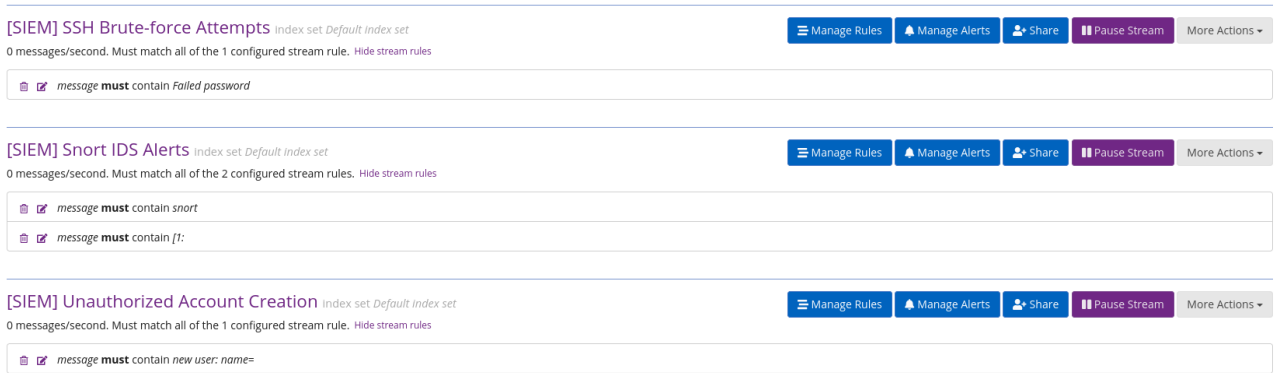


Рисунок 2.6 – Інтерфейс керування потоками у веб-консолі Graylog (продовження)

Початковий варіант правила фільтрації потоку Snort IDS Alerts, що передбачав пошук підрядка snort у тексті повідомлення, виявився недостатньо точним під час експериментальної перевірки, оскільки задовольняв також службові повідомлення системних утиліт systemctl та sudo, що згадували назву процесу snort під час його запуску чи перезапуску адміністратором, не будучи подіями виявлення атаки.

Для усунення цієї неточності правило було уточнено до пошуку характерного префікса ідентифікатора запису Snort у форматі «[1:», що присутній виключно у згенерованих підсистемою виявлення вторгнень повідомленнях про спрацювання сигнатурних правил.

Недоліком використання виключно механізму потоків є неможливість отримання уніфікованої текстової категорії події засобами стандартної візуалізації Graylog, оскільки потоки ідентифікуються внутрішніми ідентифікаторами, а не змістовними назвами. Для розв'язання цієї проблеми у роботі застосовано механізм конвеєрів обробки повідомлень (Pipelines), правила якого виконують додавання нового поля `event_category` з людиночитною назвою категорії події безпосередньо на етапі надходження повідомлення, до моменту його розподілу за потоками.

Приклад правила конвеєра для класифікації спроб підвищення привілеїв:

```
rule "Tag Privilege Escalation"
when
    contains(to_string($message.message), "pam_unix(sudo:auth)") AND
    contains(to_string($message.message), "authentication failure")
then
    set_field("event_category", "Privilege Escalation");
end
```

Застосування комбінованої умови, що враховує одночасну наявність специфічного ідентифікатора модуля автентифікації `pam_unix(sudo:auth)` та тексту повідомлення про помилку автентифікації, дозволяє уникнути хибної класифікації службових повідомлень інших підсистем, які також містять підрядок `sudo`, але не є подіями безпеки. Аналогічний принцип уточнення умов застосовано і до інших правил конвеєра. Такий підхід до нормалізації даних є ключовим для побудови гнучкої SIEM-системи, оскільки дозволяє привести різномірні повідомлення від різних систем до єдиного формату, що суттєво полегшує подальший аналіз та візуалізацію. Це суттєво пришвидшує проведення пошукових запитів та спрощує формування комплексних звітів, необхідних для оцінки загального рівня захищеності інфраструктури. Повний перелік реалізованих правил класифікації наведено у таблиці 2.6.

		Шатілов М. В.			ККІКБ.КР.Б – 125 – 12 – ПЗ	Арк.
		Воротніков В. В.				
Змн.	Арк.	№ докум.	Підпис	Дата		34

Перелік правил класифікації конвеєра Security Event Tagging

Назва правила	Значення поля event_category
Tag SSH Bruteforce	SSH Brute-force
Tag Privilege Escalation	Privilege Escalation
Tag Root Access Attempt	Direct Root Access
Tag Account Creation	Unauthorized Account Creation
Tag Cron Modification	Cron Modification
Tag SQL Injection	SQL Injection
Tag Port Scan	Port Scan
Tag Directory Scan	Directory Scan
Tag Scanner Detected	Scanner Detected

Інтерфейс редагування правил конвеєра обробки повідомлень у розділі System → Pipelines веб-консолі Graylog, що відображає перелік усіх дев'яти реалізованих правил у межах єдиного етапу обробки (Stage 0), наведено на рисунку 2.7.

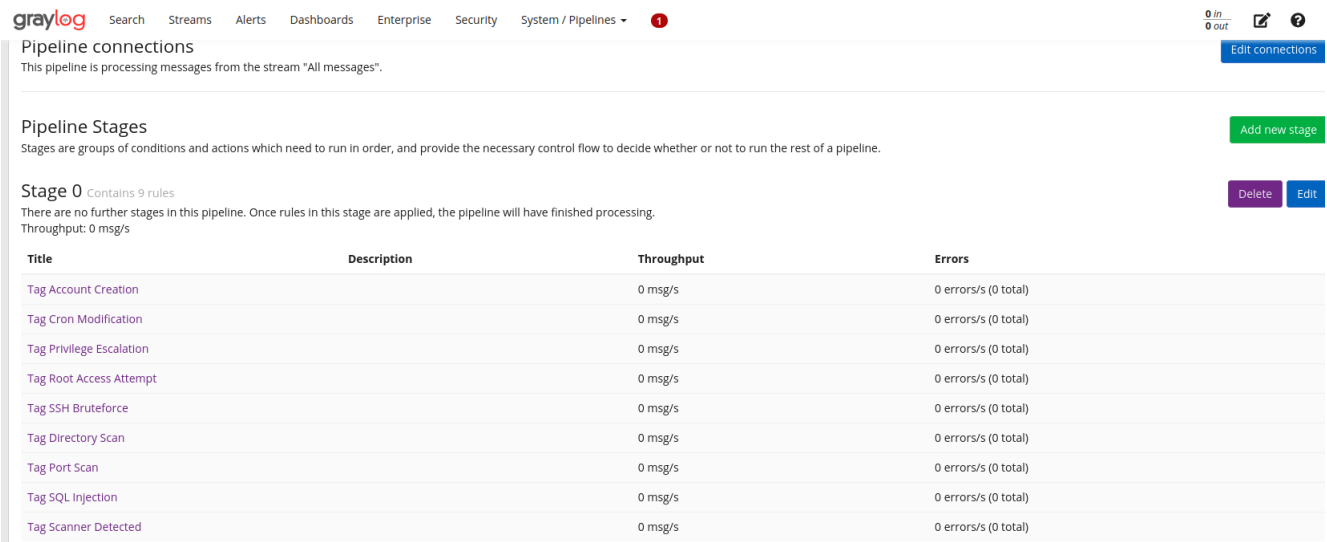


Рисунок 2.7. – Інтерфейс редагування правил конвеєра Security Event Tagging

Дев'ять реалізованих правил класифікації охоплюють усі визначені у підрозділі 2.1 варіанти використання з боку зловмисника, при цьому п'ять правил, що відповідають подіям операційної системи, застосовуються до повідомлень, які надходять до Graylog до моменту їх розподілу за потоками, описаними у таблиці 2.6, а решта чотири правила класифікують події, сформовані підсистемою виявлення

мережевих вторгнень Snort, опис якої наведено у підрозділі 2.4. Результат класифікації подій відображається на інформаційній панелі SOC Security Overview, приклад якої наведено на рисунку 2.8.

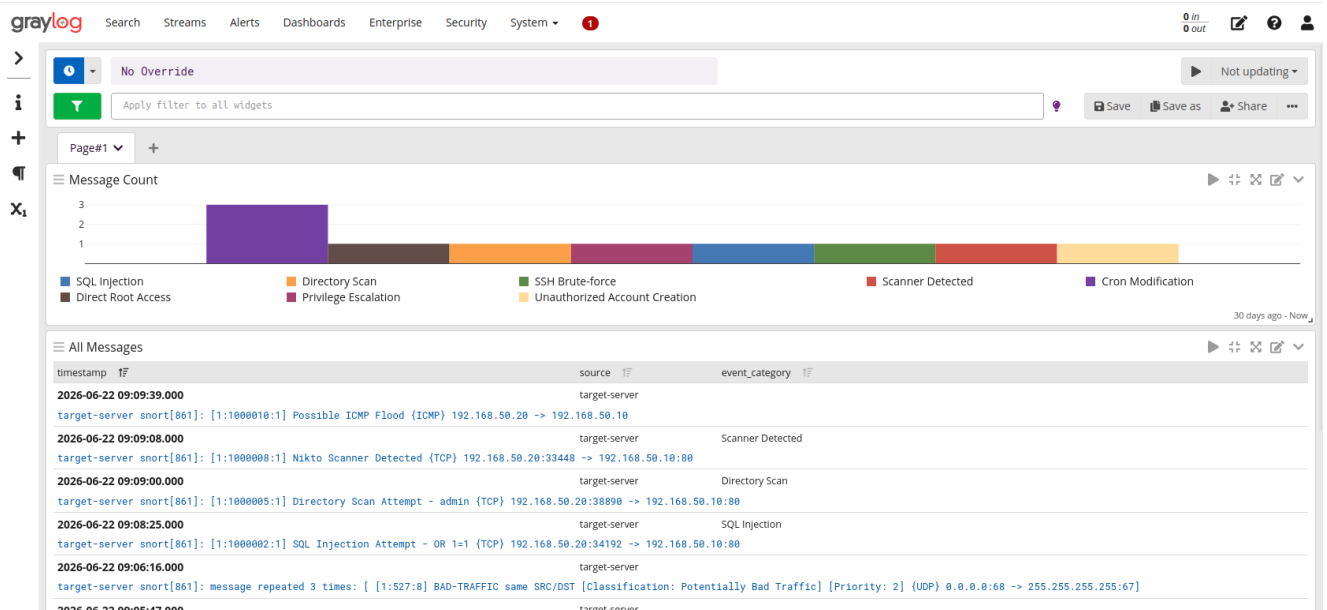


Рисунок 2.8 – Інформаційна панель SOC Security Overview у Graylog

2.4 Реалізація підсистеми виявлення мережевих вторгнень

Підсистема виявлення мережевих вторгнень реалізована на основі системи Snort, розгорнутої на цільовому вузлі інфраструктури. Аналізу підлягає мережевий інтерфейс, підключений до публічного сегмента мережі (vlan-public), оскільки саме через цей сегмент здійснюється взаємодія з потенційним зловмисником. Параметр HOME_NET конфігурації Snort встановлено відповідно до адресного простору внутрішньої мережі цільового вузла, що дозволяє коректно визначати напрямки мережевих з'єднань під час аналізу. Застосування сигнатурного аналізу у межах Snort дозволяє з високою точністю детектувати відомі вектори атак, порівнюючи мережеві пакети з базою попередньо визначених правил. Оскільки система функціонує безпосередньо на цільовому вузлі, це забезпечує мінімальну затримку при виявленні загроз, що є критично важливим для оперативного спрацювання підсистеми реагування.

Підтвердження активного стану служби Snort, отримане засобом стандартної команди керування службами systemd, наведено на рисунку 2.9.

		Шатілов М. В.			ККІКБ.КР.Б – 125 – 12 – ПЗ	Арк.
		Воротніков В. В.				
Змн.	Арк.	№ докум.	Підпис	Дата		36

```

serveradmin@target-server:~$ sudo systemctl status snort
[sudo] password for serveradmin:
• snort.service - LSB: Lightweight network intrusion detection system
   Loaded: loaded (/etc/init.d/snort; generated)
   Active: active (running) since Mon 2026-06-22 08:27:40 UTC; 28min ago
     Docs: man:systemd-sysv-generator(8)
  Process: 724 ExecStart=/etc/init.d/snort start (code=exited, status=0/SUCCESS)
    Tasks: 2 (limit: 1659)
   Memory: 89.8M (peak: 195.5M)
      CPU: 1.114s
   CGroup: /system.slice/snort.service
           └─861 /usr/sbin/snort -m 027 -D -l /var/log/snort -u snort -g snort --pid-path /run/snort/ -c /etc/snort/snort.conf -S "\$HOME_NET=[10.0.0.0/2
Jun 22 08:27:41 target-server snort[861]: Preprocessor Object: SF_IMAP Version 1.0 <Build 1>
Jun 22 08:27:41 target-server snort[861]: Preprocessor Object: apdlib Version 1.1 <Build 5>
Jun 22 08:27:41 target-server snort[861]: Preprocessor Object: SF_MQDBUS Version 1.1 <Build 1>
Jun 22 08:27:41 target-server snort[861]: Preprocessor Object: SF_SSLPP Version 1.1 <Build 4>
Jun 22 08:27:41 target-server snort[861]: Preprocessor Object: SF_SMTP Version 1.1 <Build 9>
Jun 22 08:27:41 target-server snort[861]: Preprocessor Object: SF_S7COMMPLUS Version 1.0 <Build 1>
Jun 22 08:27:41 target-server snort[861]: Preprocessor Object: SF_SIP Version 1.1 <Build 1>
Jun 22 08:27:41 target-server snort[861]: Preprocessor Object: SF_GTP Version 1.1 <Build 1>
Jun 22 08:27:41 target-server snort[861]: Preprocessor Object: SF_REPUTATION Version 1.1 <Build 1>
Jun 22 08:27:41 target-server snort[861]: Commencing packet processing (pid=861)

```

Рисунок 2.9 – Перевірка статусу служби Snort на цільовому вузлі

Для виявлення основних векторів атак на веб-застосунок розроблено набір користувацьких сигнатурних правил, перелік яких наведено у таблиці 2.7.

Таблиця 2.7

Перелік користувацьких сигнатурних правил Snort

Ідентифікатор (SID)	Назва правила	Умова виявлення
1000001–1000003	SQL Injection Attempt	Наявність конструкцій UNION SELECT, OR 1=1 у запиті
1000004	Possible NMAP Port Scan	Понад 5 SYN-пакетів від одного джерела за 5 секунд
1000005–1000007	Directory Scan Attempt	Звернення до характерних шляхів /admin, /backup, /wp-login
1000008–1000009	Scanner Detected	Підрядки Nikto, sqlmap у заголовку User-Agent
1000010	Possible ICMP Flood	Понад 10 ICMP-запитів від одного джерела за 5 секунд

Розглянемо детальніше синтаксис трьох характерних правил, що ілюструють різні механізми виявлення, застосовані у роботі. Правило виявлення спроби впровадження SQL-коду аналізує вміст пакета на наявність характерної лексичної конструкції незалежно від регістру символів.

Синтаксис правила виявлення спроби SQL-ін'єкції:

```

alert tcp any any -> $HOME_NET 80 (
  msg:"SQL Injection Attempt - UNION SELECT";
  content:"UNION"; nocase;
  content:"SELECT"; nocase; distance:0;
  sid:1000001; rev:1;
)

```

		Шатілов М. В.			ККІКБ.КР.Б – 125 – 12 – ПЗ	Арк.
		Воротніков В. В.				
Змн.	Арк.	№ докум.	Підпис	Дата		37

Параметр content визначає підрядок, наявність якого в пакеті є умовою спрацювання правила; модифікатор poscase вмикає чутливість до регістру символів, що унеможливорює обхід правила шляхом зміни регістру літер запиту. Параметр distance:0 у поєднанні з другим параметром content вказує механізму перевірки шукати підрядок SELECT у довільній позиції пакета після виявлення підрядка UNION, що дозволяє виявляти типову конструкцію SQL-ін'єкції незалежно від додаткових символів між ключовими словами запиту. Параметр sid визначає унікальний ідентифікатор правила в межах локального набору сигнатур, а параметр rev – номер редакції правила, що використовується при оновленні логіки виявлення без зміни ідентифікатора.

Правило виявлення сканування мережевих портів на відміну від розглянутого вище, не аналізує вміст пакета, а використовує механізм порогового спрацювання (threshold), вбудований у саму систему Snort.

Синтаксис правила виявлення сканування портів:

```
alert tcp any any -> $HOME_NET any (
  msg:"Possible NMAP Port Scan";
  flags:S;
  threshold: type threshold, track by_src,
              count 5, seconds 5;
  sid:1000004; rev:1;
)
```

Параметр flags:S обмежує застосування правила лише до пакетів із встановленим прапором SYN, що відповідає початковій стадії встановлення TCP-з'єднання та є характерною ознакою інструментів мережевого сканування, які зазвичай не завершують повне тристороннє узгодження з'єднання. Блок threshold реалізує підрахунок кількості пакетів, що задовольняють умову правила, у межах визначеного часового вікна: значення track by_src вказує на групування підрахунку за адресою джерела пакета, а пара параметрів count 5, seconds 5 визначає умову спрацювання – не менше п'яти пакетів з прапором SYN від однієї адреси джерела протягом п'яти секунд. Застосування вбудованого механізму порогового

		Шатілов М. В.			ККІКБ.КР.Б – 125 – 12 – ПЗ	Арк.
		Воротніков В. В.				
Змн.	Арк.	№ докум.	Підпис	Дата		38

спрацювання на рівні самої системи Snort, а не на рівні програмного модуля автоматизованого реагування, дозволяє зменшити навантаження на канал передавання журналів подій, оскільки повідомлення про сканування формується одноразово після досягнення порогового значення, а не для кожного окремого пакета.

Правило виявлення спроби сканування каталогів веб-застосунку ілюструє найпростіший різновид сигнатурного аналізу – пошук характерного підрядка у складі URI запиту.

Синтаксис правила виявлення спроби сканування каталогу /admin:

```
alert tcp any any -> $HOME_NET 80 (  
  msg:"Directory Scan Attempt - admin";  
  content:"/admin"; nocase;  
  sid:1000005; rev:1;  
)
```

Аналогічні за структурою правила розроблено для виявлення звернень до шляхів /backup та /wp-login (ідентифікатори 1000006 та 1000007), що відповідають типовим цілям автоматизованого сканування вразливостей веб-застосунків. Правила виявлення сканерів вразливостей застосовують модифікатор http_header замість стандартного пошуку в усьому пакеті, що обмежує область пошуку підрядків Nikto та sqlmap виключно заголовками HTTP-запиту, зокрема полем User-Agent, у якому зазначені засоби автоматизованого сканування зазначають власну ідентифікаційну інформацію за умовчанням.

Синтаксис правила виявлення сканера вразливостей Nikto:

```
alert tcp any any -> $HOME_NET 80 (  
  msg:"Nikto Scanner Detected";  
  content:"Nikto"; http_header;  
  sid:1000008; rev:1;  
)
```

Останнє з реалізованих користувацьких правил (ідентифікатор 1000010) призначене для виявлення надмірної кількості ICMP-запитів від одного джерела, що може свідчити про спробу дослідження доступності вузла або найпростіший різно-

		Шатілов М. В.			ККІКБ.КР.Б – 125 – 12 – ПЗ	Арк.
		Воротніков В. В.				
Змн.	Арк.	№ докум.	Підпис	Дата		39

вид атаки на відмову в обслуговуванні мережевого рівня. Правило застосовує модифікатор `ittype:8` для обмеження аналізу виключно запитами типу Echo Request (стандартними запитами утиліти `ping`) у поєднанні з механізмом порогового спрацювання, аналогічним до розглянутого вище правила виявлення сканування портів, з пороговим значенням десять пакетів за п'ять секунд.

Результати аналізу мережевого трафіку записуються підсистемою Snort у вигляді повідомлень системного журналу (Syslog), які надходять до Graylog за тим самим механізмом, що й журнали операційної системи, описаним у підрозділі 2.3. Структурна схема обробки мережевого пакета – від моменту його надходження на інтерфейс цільового вузла до формування класифікованої події у Graylog – наведена на рисунку 2.10.

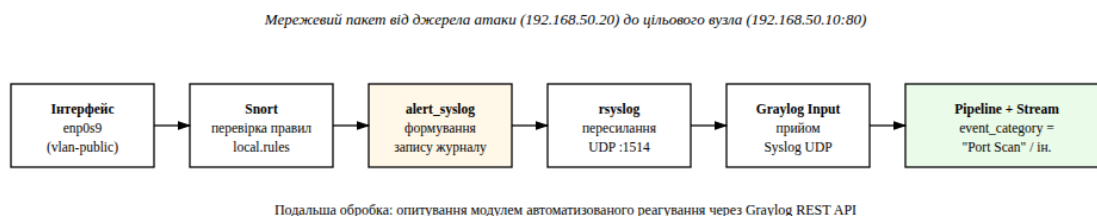


Рисунок 2.10 – Схема обробки мережевого пакета підсистемою виявлення вторгнень

Під час розроблення правил виявлено особливість, що потребує врахування при інтерпретації результатів: окремі автоматизовані засоби сканування (зокрема Nikto) одночасно задовольняють умови кількох незалежних правил – Directory Scan Attempt, Possible NMAP Port Scan та Scanner Detected, оскільки генерують значну кількість запитів до характерних шляхів за короткий проміжок часу. Така властивість не є недоліком реалізації, а демонструє стійкість системи до різних інструментів сканування завдяки одночасному спрацюванню кількох незалежних сигнатур, що підвищує ймовірність своєчасного виявлення атаки.

		Шатілов М. В.			ККІКБ.КР.Б – 125 – 12 – ПЗ	Арк.
		Воротніков В. В.				
Змн.	Арк.	№ докум.	Підпис	Дата		40

2.5. Компонентна будова та реалізація програмного модуля автоматизованого реагування

Підсистема автоматизованого реагування на інциденти реалізована у вигляді окремого програмного модуля мовою Python, що функціонує на сервері SIEM/SOC незалежно від інших компонентів комплексу. Модуль складається з чотирьох функціональних компонентів: компонента опитування програмного інтерфейсу Graylog, компонента аналізу порогових значень, компонента віддаленого блокування мережевих адрес та компонента взаємодії з месенджером Telegram. Компонентна структура модуля наведена на рисунку 2.11, повний текст програмного коду наведено у Додатку В.

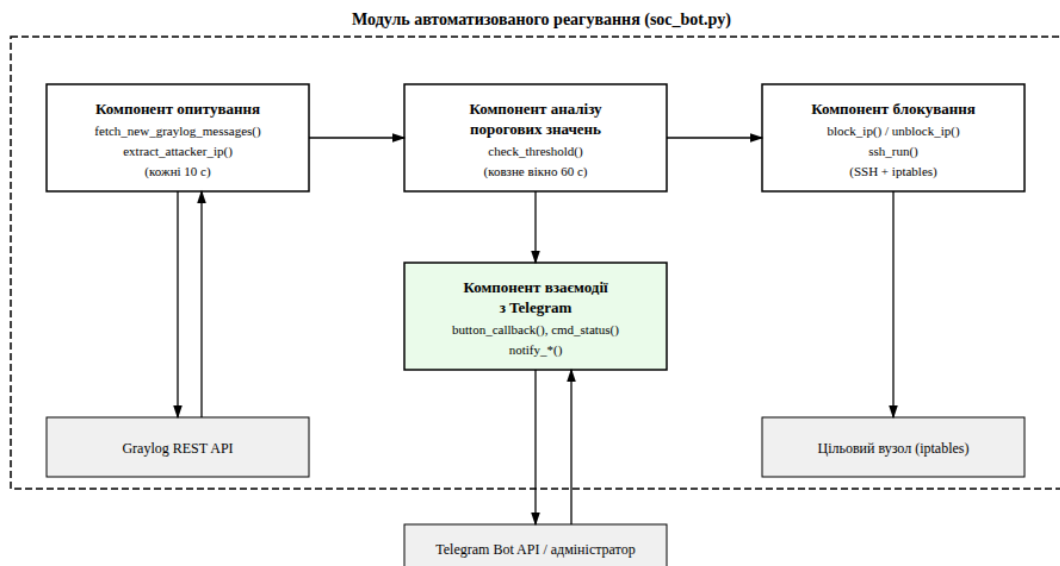


Рисунок 2.11 – Компонентна структура модуля автоматизованого реагування

Для візуалізації цілісності закладеної логіки та послідовності прийняття рішень під час обробки інцидентів, на рисунку 2.12 наведено алгоритм функціонування підсистеми автоматизованого реагування. Цей алгоритм демонструє повний цикл обробки: від отримання класифікованої події з платформи Graylog до прийняття рішення про автоматичне блокування з урахуванням перевірки довірених IP-адрес, оцінки порогових значень та механізму захисту від розподілених атак на відмову в обслуговуванні.

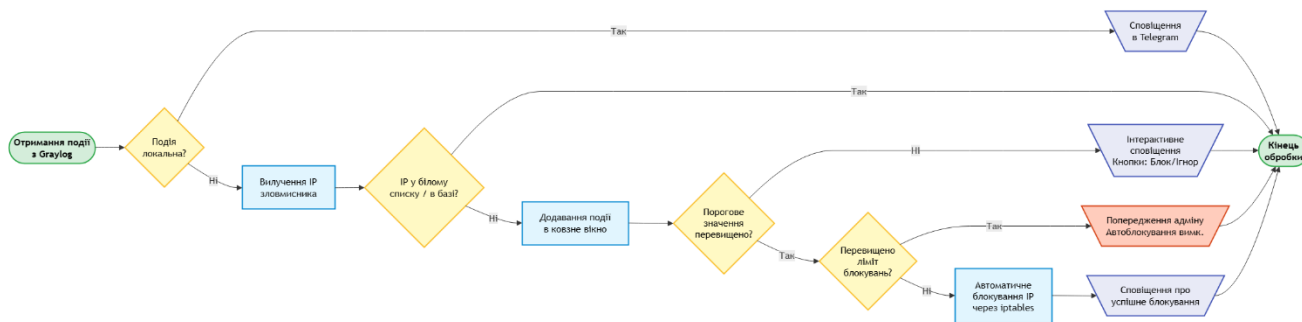


Рисунок 2.12 – Алгоритм функціонування підсистеми автоматизованого реагування

Компонент опитування виконує періодичний, з інтервалом 10 секунд, запит до програмного інтерфейсу Graylog REST API з фільтром за наявністю поля `event_category`, сформованого конвеєром обробки повідомлень, описаним у підрозділі 2.3. Запит виконується до кінцевої точки `/api/search/universal/relative` із параметром `query` зі значенням `_exists_:event_category`, що повертає лише ті повідомлення, які вже пройшли класифікацію засобами конвеєра Security Event Tagging, та параметром `range`, що обмежує вибірку подіями за останню хвилину з метою уникнення повторної обробки застарілих записів.

Фрагмент програмного коду функції опитування Graylog REST API:

```

def fetch_new_graylog_messages():
    url = f"{config.GRAYLOG_URL}/api/search/universal/relative"
    params = {
        "query": "_exists_:event_category",
        "range": 60,
        "limit": 50,
        "sort": "timestamp:desc",
    }
    resp = requests.get(
        url, params=params,
        auth=(config.GRAYLOG_API_TOKEN, "token"),
        headers={"Accept": "application/json"}, timeout=10,
    )
    resp.raise_for_status()
    return resp.json().get("messages", [])
  
```

Автентифікація запиту здійснюється за допомогою персонального токена доступу, сформованого засобами інтерфейсу керування користувачами Graylog, що дозволяє уникнути передавання облікових даних адміністратора платформи у програмному коді модуля. Для кожної отриманої події визначається мережева адреса джерела, що становить окрему технічну складність, оскільки різні підсистеми формують повідомлення у різних форматах. Записи підсистеми Snort містять мережеву адресу джерела атаки безпосередньо у тексті повідомлення у форматі «адреса:порт -> адреса:порт», у той час як стандартне поле `gl2_remote_ip`, що формується вхідним модулем Graylog, у цьому випадку відображає адресу цільового вузла – джерела журналу подій, а не адресу зловмисника, що було виявлено експериментально під час тестування модуля та враховано в реалізації функції визначення адреси.

Фрагмент програмного коду функції визначення адреси джерела атаки:

```
def extract_attacker_ip(message_text, fallback_ip):
    pattern = r"\b(?:\d{1,3}\.){3}\d{1,3}(?:\:\d+)?\s*->\s*"
    match = re.search(pattern, message_text)
    if match:
        return match.group(1)
    return fallback_ip
```

Наведена функція повертає першу мережеву адресу, що передуює характерній послідовності символів «->» у тексті повідомлення, що відповідає форматуванню журналів підсистеми Snort. Для подій операційної системи, що не передбачають мережевої взаємодії (модифікація завдань cron, створення облікових записів, локальне підвищення привілеїв), компонент відносить подію до категорії локальних подій, для яких процедура блокування не застосовується, а виконується лише інформаційне сповіщення адміністратора. Визначена мережева адреса додатково перевіряється за переліком довірених адрес інфраструктури (білим списком); подія, джерелом якої є довірена адреса, виключається з подальшої обробки з метою унеможливлення випадкового блокування власних вузлів інфраструктури під час виконання штатних адміністративних дій.

		Шатілов М. В.			ККІКБ.КР.Б – 125 – 12 – ПЗ	Арк.
		Воротніков В. В.				43
Змн.	Арк.	№ докум.	Підпис	Дата		

Компонент аналізу порогових значень реалізує ключову логіку ухвалення рішення щодо способу реагування на подію. Для кожної комбінації «мережева адреса – категорія події» модуль накопичує часову позначку кожного спрацювання у структурі даних типу deque, що реалізує ефективне додавання нових та вилучення застарілих елементів з обох кінців послідовності. Перевірка досягнення порогового значення виконується функцією.

Фрагмент програмного коду функції перевірки порогового значення:

```
def check_threshold(ip: str, category: str) -> bool:
    threshold = config.THRESHOLDS.get(category)
    if not threshold:
        return False
    key = (ip, category)
    now = time.time()
    history = event_history[key]
    history.append(now)
    window = threshold["window"]
    while history and now - history[0] > window:
        history.popleft()
    return len(history) >= threshold["count"]
```

Застосований підхід реалізує механізм ковзного часового вікна (sliding window): під час кожного звернення до функції з накопиченої історії спрацювань вилучаються записи, час яких перевищує визначене вікно спостереження, після чого порівнюється кількість записів, що залишилися, із пороговим значенням, визначеним для відповідної категорії події у конфігурації модуля. Якщо кількість подій у межах вікна досягає або перевищує цей порог, модуль ухвалює рішення про автоматичне блокування мережевої адреси; в іншому випадку адміністратору надсилається інтерактивне сповіщення з можливістю самостійного ухвалення рішення. Вибір порогових значень є компромісом між чутливістю виявлення та ризиком хибних спрацювань. Наведені параметри враховують як специфіку сигнатур, так і очікувану інтенсивність атак для нашого тестового середовища. Перелік порогових значень, обґрунтування яких наведено у третьому розділі роботи, представлено у таблиці 2.8.

		Шатілов М. В.			ККІКБ.КР.Б – 125 – 12 – ПЗ	Арк.
		Воротніков В. В.				
Змн.	Арк.	№ докум.	Підпис	Дата		44

Порогові значення автоматичного блокування за категоріями подій

Категорія події	Поріг автоблокування
SQL Injection	3 події за 60 секунд
Port Scan	1 подія за 60 секунд
Directory Scan	5 подій за 60 секунд
Scanner Detected	1 подія за 60 секунд
SSH Brute-force	5 подій за 60 секунд
Privilege Escalation	3 події за 60 секунд (локальна подія, без блокування)
Direct Root Access	3 події за 60 секунд (локальна подія, без блокування)
Unauthorized Account Creation	1 подія (локальна подія, без блокування)
Cron Modification	1 подія (локальна подія, без блокування)

Компонент віддаленого блокування виконує безпосереднє застосування правила фільтрації трафіку на цільовому вузлі шляхом віддаленого виконання команди утиліти iptables через захищене з'єднання SSH з використанням механізму автентифікації за криптографічним ключем без застосування пароля [22]. Допоміжна функція виконання віддаленої команди інкапсулює деталі встановлення SSH-з'єднання та обробку можливих помилок мережевої взаємодії.

Фрагмент програмного коду допоміжної функції виконання віддаленої команди SSH:

```
def ssh_run(remote_command: str):
    cmd = [
        "ssh", "-i", config.SSH_KEY_PATH,
        "-o", "StrictHostKeyChecking=no",
        "-o", "ConnectTimeout=5",
        f"{config.TARGET_USER}@{config.TARGET_HOST}",
        remote_command,
    ]
    try:
        result = subprocess.run(
            cmd, capture_output=True,
            text=True, timeout=15,
```

		Шатілов М. В.			ККІКБ.КР.Б – 125 – 12 – ПЗ	Арк.
		Воротніков В. В.				
Змн.	Арк.	№ докум.	Підпис	Дата		45

```

)
if result.returncode == 0:
    return True, result.stdout.strip()
return False, result.stderr.strip()
except Exception as e:
    return False, str(e)

```

Параметр ConnectTimeout обмежує час очікування встановлення мережевого з'єднання п'ятьма секундами, а параметр timeout функції subprocess.run обмежує загальний час виконання команди п'ятнадцятьма секундами, що унеможливорює блокування основного потоку виконання модуля у разі тимчасової недоступності цільового вузла мережею або зависання віддаленої команди. Параметр StrictHostKeyChecking, встановлений у значення no, вимикає інтерактивну перевірку відбитка ключа SSH-сервера, що є прийнятним в умовах ізолюваного тестового стенда з обмеженою та контрольованою кількістю вузлів, але потребує окремого розгляду під час можливого впровадження комплексу в межах реальної корпоративної мережі. Функція повертає кортеж зі значенням успішності виконання та текстом виведення або помилки, що дозволяє компоненту блокування коректно відобразити причину невдачі в журналі роботи модуля без переривання основного циклу опитування. Фрагмент програмного коду власне функції блокування мережевої адреси, що використовує описану допоміжну функцію.

Фрагмент програмного коду функції блокування мережевої адреси:

```

def block_ip(ip: str) -> bool:
    success, output = ssh_run(
        f"sudo iptables -C INPUT -s {ip} -j DROP || "
        f"sudo iptables -I INPUT -s {ip} -j DROP"
    )
    if success:
        unblock_time = datetime.now() + timedelta(seconds=config.BLOCK_DURATION)
        blocked_ips[ip] = unblock_time
        recent_blocks.append(datetime.now())
    return success

```

		Шатілов М. В.			ККІКБ.КР.Б – 125 – 12 – ПЗ	Арк.
		Воротніков В. В.				
Змн.	Арк.	№ докум.	Підпис	Дата		46

Застосована у наведеній функції конструкція командного рядка iptables -C ... || iptables -I ... реалізує ідемпотентність операції блокування: команда -C (check) перевіряє наявність правила в таблиці фільтрації без внесення змін, і лише за умови його відсутності (ненульовий код завершення) виконується команда -I (insert) додавання нового правила. Такий підхід унеможливорює створення дублікатів правила для однієї й тієї ж мережевої адреси у разі повторного виявлення подій від уже заблокованого джерела до моменту оновлення внутрішнього стану модуля.

Передумовою застосування блокування є перевірка мережевої адреси за переліком довірених адрес (білим списком), до якого включено адреси сервера SIEM/SOC та робочої станції адміністратора з метою унеможливлення випадкового блокування власних вузлів інфраструктури. Додатково реалізовано захист від масового блокування: якщо кількість нових блокувань протягом 60 секунд перевищує визначене граничне значення, що може свідчити про розподілену атаку типу відмови в обслуговуванні [21], модуль припиняє виконання подальших автоматичних блокувань та надсилає адміністратору відповідне попередження, унеможливаючи неконтрольоване перевантаження таблиці правил міжмережевого екрана цільового вузла. Кожне автоматичне блокування має обмежену тривалість дії, після завершення якої окремий фоновий процес модуля виконує автоматичне розблокування мережевої адреси.

Компонент взаємодії з месенджером Telegram реалізований із застосуванням бібліотеки python-telegram-bot [20] та забезпечує двонапрямлену комунікацію з адміністратором безпеки. У разі автоматичного блокування адміністратору надсилається інформаційне повідомлення з інтерактивною кнопкою скасування блокування. У разі, якщо подія не досягла порогового значення, повідомлення містить дві кнопки – підтвердження блокування та ігнорування події, натискання яких обробляється відповідним функціональним обробником зворотного виклику (callback handler). Це дозволяє адміністратору миттєво ухвалювати рішення щодо нейтралізації загроз безпосередньо з інтерфейсу месенджера, що суттєво підвищує оперативність реагування. Приклад інтерактивного сповіщення наведено на рисунку 2.13.

		Шатілов М. В.			ККІКБ.КР.Б – 125 – 12 – ПЗ	Арк.
		Воротніков В. В.				
Змн.	Арк.	№ докум.	Підпис	Дата		47

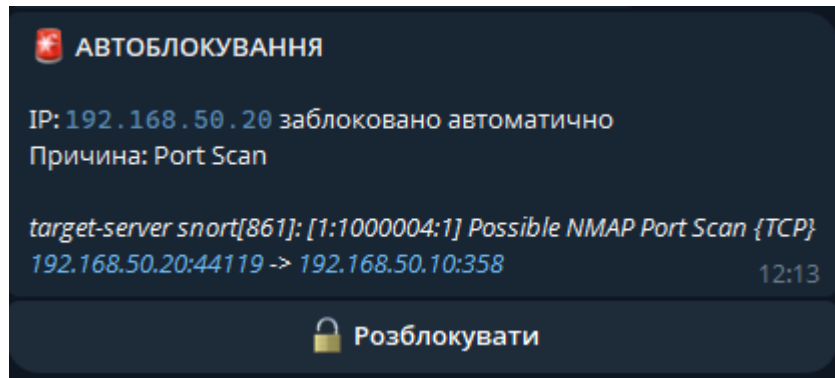


Рисунок 2.13 – Приклад інтерактивного сповіщення в месенджері Telegram

Формування інтерактивних кнопок здійснюється засобами класу `InlineKeyboardMarkup` бібліотеки `python-telegram-bot`, кожна кнопка якого пов'язується з рядком `callback_data`, що кодує тип дії та ідентифікатор події. Обробка натискання кнопки виконується єдиним функціональним обробником.

Фрагмент програмного коду обробника натискання інтерактивних кнопок:

```
def button_callback(update, context):
    query = update.callback_query
    query.answer()
    data = query.data

    if data.startswith("unblock:"):
        ip = data.split(":", 1)[1]
        if unblock_ip(ip):
            query.edit_message_text(
                f"✅ IP {ip} розблоковано вручну.")

    elif data.startswith("block:"):
        pid = int(data.split(":", 1)[1])
        action = pending_actions.pop(pid, None)
        if action and block_ip(action["ip"]):
            query.edit_message_text(
                f"🔒 IP {action['ip']} заблоковано вручну.")

    elif data.startswith("ignore:"):
        pid = int(data.split(":", 1)[1])
        pending_actions.pop(pid, None)
```

		Шатілов М. В.			ККІКБ.КР.Б – 125 – 12 – ПЗ	Арк.
		Воротніков В. В.				
Змн.	Арк.	№ докум.	Підпис	Дата		48

```
query.edit_message_text("✅ Подію проігноровано.")
```

Префікс рядка `callback_data` (`unblock`, `block` або `ignore`) визначає тип дії, що має бути виконана, а частина рядка після символу двокрапки кодує ідентифікатор мережевої адреси або внутрішній числовий ідентифікатор події, очікуваної на розгляд адміністратора (`pending_id`). Останній використовується для подій, що ще не досягли порогового значення автоблокування: кожній такій події присвоюється унікальний послідовний ідентифікатор, що зберігається у словнику `pending_actions` разом із мережевою адресою та категорією події до моменту ухвалення адміністратором рішення, після чого відповідний запис видаляється зі словника незалежно від ухваленого рішення з метою запобігання повторній обробці того ж запиту. Виклик методу `query.answer()` є обов'язковим технічним викликом програмного інтерфейсу Telegram Bot API, що підтверджує отримання натискання кнопки клієнтським застосунком та запобігає відображенню індикатора очікування відповіді в інтерфейсі месенджера.

Окрім обробки натискань інтерактивних кнопок, модуль підтримує текстові команди адміністратора. Команда `/status` повертає перелік мережевих адрес, заблокованих на поточний момент, із зазначенням часу автоматичного розблокування, а команда `/unblock` з параметром у вигляді мережевої адреси дозволяє виконати позачергове розблокування.

Фрагмент програмного коду обробника команди `/status`:

```
def cmd_status(update, context):
    if not blocked_ips:
        update.message.reply_text(
            "Наразі немає заблокованих IP.")
        return
    lines = ["🔒 Заблоковані IP:\n"]
    for ip, until in blocked_ips.items():
        if until:
            lines.append(
                f"• {ip} – до "
                f"{until.strftime('%Y-%m-%d %H:%M:%S')}")
        else:
```

		Шатілов М. В.			ККІКБ.КР.Б – 125 – 12 – ПЗ	Арк.
		Воротніков В. В.				
Змн.	Арк.	№ докум.	Підпис	Дата		49

```
lines.append(f"• {ip} – назавжди")
update.message.reply_text("\n".join(lines))
```

Реєстрація обробників команд та натискань кнопок виконується під час ініціалізації модуля засобами диспетчера бібліотеки python-telegram-bot, що пов'язує текстову команду або шаблон рядка callback_data із відповідною функцією обробки. Опитування програмного інтерфейсу Telegram Bot API щодо нових команд адміністратора та обробка подій Graylog виконуються у двох незалежних фонових потоках виконання (threading.Thread), що працюють паралельно з основним циклом програми та дозволяють модулю одночасно реагувати на нові інциденти безпеки та оперативно обробляти команди адміністратора без взаємного блокування.

Окремої уваги потребує забезпечення стійкості модуля до помилок мережевої взаємодії, що є особливо важливим з огляду на залежність функціонування модуля від доступності трьох зовнішніх програмних інтерфейсів – Graylog REST API, Telegram Bot API та протоколу SSH до цільового вузла. Кожен мережевий запит до зазначених інтерфейсів виконується з обмеженням часу очікування відповіді (timeout) та обробкою можливих винятків, що унеможливорює зависання основного циклу опитування у разі тимчасової недоступності одного з компонентів інфраструктури. Узагальнений перелік основних параметрів конфігурації модуля, що визначають його поведінку, наведено у таблиці 2.9.

Таблиця 2.9

Перелік основних параметрів конфігурації модуля автоматизованого реагування

Параметр конфігурації	Значення	Призначення
POLL_INTERVAL	10 секунд	Періодичність опитування Graylog REST API
BLOCK_DURATION	3600 секунд	Тривалість дії автоматичного блокування мережевої адреси
MAX_BLOCKS_PER_WINDOW	10 блокувань	Граничне число нових блокувань для активації захисту від масового блокування

Параметр конфігурації	Значення	Призначення
MAX_BLOCKS_WINDOW_SECONDS	60 секунд	Часове вікно для підрахунку кількості нових блокувань
WHITELIST	перелік адрес	Мережеві адреси, що виключаються з процедури блокування

Виокремлення зазначених параметрів у окремий конфігураційний файл, відокремлений від основної логіки модуля, відповідає принципу розділення конфігурації від програмного коду та дозволяє адміністратору коригувати поведінку системи (зокрема, чутливість виявлення та тривалість блокування) без необхідності внесення змін до самого алгоритму обробки подій. Реалізована компонентна структура модуля забезпечує повний цикл обробки інциденту безпеки – від отримання класифікованої події з SIEM-платформи до виконання коригувальної дії та документування результату реагування, що відповідає визначеним у підрозділі 2.1 варіантам використання системи.

Висновки до другого розділу

У результаті виконання проектної частини роботи розроблено та реалізовано архітектуру тестового стенда із трисегментною мережевою топологією, що забезпечує безпечне відтворюване моделювання мережевих атак при повній ізоляції сегмента управління від публічного сегмента. Визначено варіанти використання системи з боку адміністратора безпеки та потенційного зловмисника, що стали основою для формування функціональних вимог до окремих підсистем комплексу.

Реалізовано підсистему моніторингу інфраструктурних метрик на основі Prometheus з набором правил генерації тривог для шести критичних показників стану цільового вузла. Реалізовано підсистему централізованого аналізу подій безпеки на основі Graylog із застосуванням механізмів потоків та конвеєрів обробки повідомлень, що дозволило виконати точну класифікацію подій безпеки за дев'ятьма категоріями безпосередньо на етапі їх надходження до системи.

		Шатілов М. В.			ККІКБ.КР.Б – 125 – 12 – ПЗ	Арк.
		Воротніков В. В.				
Змн.	Арк.	№ докум.	Підпис	Дата		51

Реалізовано підсистему виявлення мережеских вторгнень на основі Snort із набором користувацьких сигнатурних правил, що охоплюють основні вектори атак на веб-застосунки – SQL-ін’єкції, сканування портів та каталогів, використання автоматизованих засобів сканування вразливостей. Розроблено та реалізовано програмний модуль автоматизованого реагування на інциденти, що поєднує аналіз порогових значень спрацювання, віддалене блокування мережеских адрес засобами міжмережевого екрана цільового вузла та інтерактивну взаємодію з адміністратором безпеки через месенджер Telegram.

Отримана програмна реалізація забезпечує повний цикл обробки інциденту безпеки – від виявлення підозрілої активності засобами моніторингу метрик, аналізу журналів подій або мережевого трафіку до автоматизованого або напівавтоматизованого реагування на загрозу. Результати тестування працездатності реалізованого програмного комплексу, опис інтерфейсу та порядку його налаштування наведено у третьому розділі роботи.

		Шатілов М. В.			ККІКБ.КР.Б – 125 – 12 – ПЗ	Арк.
		Воротніков В. В.				
Змн.	Арк.	№ докум.	Підпис	Дата		52

РОЗДІЛ 3 ОПИС ІНТЕРФЕЙСУ, ПОРЯДОК НАЛАШТУВАННЯ ТА ТЕСТУВАННЯ ПРОГРАМНОГО КОМПЛЕКСУ

3.1 Порядок встановлення та налаштування програмного комплексу

Розгортання розробленого програмного комплексу виконується у визначеній послідовності, що враховує залежності між компонентами трьох віртуальних машин тестового стенда. Загальний порядок встановлення охоплює шість послідовних етапів: підготовку віртуальних машин засобами VirtualBox, налаштування мережевих інтерфейсів усіх вузлів, встановлення та конфігурацію контейнеризованих сервісів сервера SIEM/SOC, встановлення агентів моніторингу та підсистеми виявлення вторгнень на цільовому вузлі, налаштування взаємодії між компонентами (потоків, конвеєрів обробки, вхідних модулів) та розгортання програмного модуля автоматизованого реагування.

Підготовка віртуальних машин передбачає створення трьох екземплярів у середовищі VirtualBox з мінімальними апаратними характеристиками, визначеними у підрозділі 1.4, та підключення відповідних мережевих адаптерів до сегментів, описаних у підрозділі 2.2. Для сервера SIEM/SOC та цільового вузла застосовано дистрибутив Ubuntu Server без графічного інтерфейсу користувача, що зменшує споживання апаратних ресурсів та відповідає типовій практиці розгортання серверних компонентів інфраструктури.

Після встановлення базової операційної системи на сервері SIEM/SOC виконується оновлення пакетів та встановлення Docker Compose за допомогою послідовності команд.

Послідовність команд встановлення Docker та Docker Compose на сервері SIEM/SOC:

```
sudo apt update && sudo apt upgrade -y
sudo apt install -y ca-certificates curl gnupg
sudo install -m 0755 -d /etc/apt/keyrings
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | \
    sudo gpg --dearmor -o /etc/apt/keyrings/docker.gpg
echo "deb [signed-by=/etc/apt/keyrings/docker.gpg] \
    https://download.docker.com/linux/ubuntu \
```

		Шатілов М. В.			ККІКБ.КР.Б – 125 – 12 – ПЗ	Арк.
		Воротніков В. В.				
Змн.	Арк.	№ докум.	Підпис	Дата		53

```
$(. /etc/os-release && echo $VERSION_CODENAME) stable" | \
sudo tee /etc/apt/sources.list.d/docker.list
sudo apt update
sudo apt install -y docker-ce docker-ce-cli \
containerd.io docker-compose-plugin
```

Наведена послідовність команд додає офіційний криптографічний ключ та репозиторій постачальника Docker до списку джерел пакетів операційної системи, що забезпечує встановлення останньої стабільної версії середовища контейнеризації, на відміну від застарілої версії, що міститься у стандартному репозиторії дистрибутиву Ubuntu.

Розгортання контейнеризованих сервісів виконується одною командою `docker compose up -d`, що використовує конфігураційний файл `docker-compose.yml`, розглянутий у підрозділі 2.2, для послідовного створення та запуску шести сервісів з урахуванням визначених залежностей (параметр `depends_on` забезпечує запуск сервісів `mongodb` та `opensearch` до початку ініціалізації сервісу `graylog`). Перевірку успішності запуску всіх контейнерів виконують командою.

Команда перевірки статусу контейнеризованих сервісів:

```
docker ps --format "table {{.Names}}\t{{.Status}}"
```

Очікуваним результатом виконання наведеної команди є відображення усіх шести сервісів зі статусом `Up`, а для сервісу `graylog` – додатково позначки (`healthy`) після завершення внутрішньої перевірки готовності, що зазвичай триває від двох до п'яти хвилин після першого запуску. Під час первинного розгортання необхідно виконати початкове налаштування облікового запису адміністратора Graylog через веб-інтерфейс за адресою `http://10.0.0.20:9000`, створити вхідний модуль прийому журналів типу Syslog UDP на порту 1514, а також сформувати персональний токен доступу до програмного інтерфейсу REST API через розділ `System → Users and Teams → Edit tokens`, що використовується програмним модулем автоматизованого реагування для автентифікації запитів.

На цільовому вузлі послідовність налаштування охоплює встановлення веб-сервера Nginx, агентів Node Exporter та Nginx Prometheus Exporter, системи вияв-

		Шатілов М. В.			ККІКБ.КР.Б – 125 – 12 – ПЗ	Арк.
		Воротніков В. В.				
Змн.	Арк.	№ докум.	Підпис	Дата		54

лення вторгнень Snort з розробленим набором користувацьких правил, а також конфігурацію служби rsyslog для пересилання журналів подій на сервер SIEM/SOC. Встановлення системи Snort та визначення мережевого інтерфейсу аналізу виконується послідовністю команд.

Послідовність команд встановлення та налаштування підсистеми Snort:

```
sudo apt install snort -y
# під час встановлення вказати мережу 10.0.0.0/24
sudo nano /etc/snort/snort.debian.conf
# DEBIAN_SNORT_INTERFACE="enp0s9"
sudo nano /etc/snort/rules/local.rules
# додати користувацькі правила (підрозділ 2.4)
sudo systemctl restart snort
sudo systemctl status snort
```

Завершальним етапом є генерація пари криптографічних ключів SSH на сервері SIEM/SOC та копіювання публічного ключа на цільовий вузол, що забезпечує можливість виконання програмним модулем автоматизованого реагування віддалених команд блокування без інтерактивного введення пароля, а також надання обмеженого дозволу виконання команди iptables від імені суперкористувача без запиту пароля.

Послідовність налаштування безпарольного доступу SSH та обмежених прав sudo:

```
# на сервері SIEM/SOC
ssh-keygen -t ed25519 -f ~/.ssh/soc_key -N ""
ssh-copy-id -i ~/.ssh/soc_key serveradmin@10.0.0.10

# на цільовому вузлі (sudo visudo)
serveradmin ALL=(ALL) NOPASSWD: /usr/sbin/iptables
```

Застосування обмеженого дозволу NOPASSWD виключно для команди iptables, а не для довільної команди від імені суперкористувача, відповідає принципу найменших привілеїв та обмежує потенційний вплив компрометації програм-

		Шатілов М. В.			ККІКБ.КР.Б – 125 – 12 – ПЗ	Арк.
		Воротніков В. В.				
Змн.	Арк.	№ докум.	Підпис	Дата		55

ного модуля на інфраструктуру цільового вузла – навіть у разі зловживання обліковим записом serveradmin зловмисник матиме можливість виконувати лише операції фільтрації мережевого трафіку, а не довільні системні команди.

Запуск програмного модуля автоматизованого реагування виконується після встановлення залежних бібліотек засобом менеджера пакетів `pip` та заповнення конфігураційного файлу `config.py` реальними значеннями токенів доступу, адрес вузлів та порогових значень, визначених у таблиці 2.8, послідовністю команд.

Послідовність команд встановлення залежностей та запуску програмного модуля:

```
pip3 install python-telegram-bot==13.15 \
    requests --break-system-packages
pip3 install "urllib3<2.0" --break-system-packages \
    --force-reinstall
python3 soc_bot.py
```

Фіксація версії бібліотеки `urllib3` нижче другої основної версії є необхідною технічною вимогою, виявленою експериментально під час розгортання модуля: бібліотека `python-telegram-bot` версії 13.15 використовує застарілий внутрішній модуль роботи з HTTP-запитами, що конфліктує з новішими версіями `urllib3`, встановленими за умовчанням як залежність бібліотеки `requests` у середовищі Python 3.12. Успішний запуск підтверджується надходженням початкового сповіщення «SOC Bot запущено та готовий до моніторингу» в месенджер Telegram та появою відповідних записів у журналі виконання модуля в консолі термінала.

3.2 Структура та основні елементи інтерфейсу програмного комплексу

Програмний комплекс не передбачає розроблення власного графічного інтерфейсу користувача, натомість використовує веб-інтерфейси складових компонентів (Grafana, Graylog) та інтерфейс месенжера Telegram як засоби взаємодії з адміністратором безпеки, що відповідає сучасній практиці побудови розподілених систем моніторингу з повторним використанням інтерфейсів спеціалізованих компонентів замість розроблення вузькоспеціалізованого застосунку.

		Шатілов М. В.			ККІКБ.КР.Б – 125 – 12 – ПЗ	Арк.
		Воротніков В. В.				
Змн.	Арк.	№ докум.	Підпис	Дата		56

Основним елементом інтерфейсу підсистеми моніторингу інфраструктурних метрик є дашборд Node Exporter Full у Grafana, що відображає графіки завантаження центрального процесора, використання оперативної пам'яті, дискового простору та мережевого трафіку цільового вузла в реальному часі з можливістю вибору довільного часового діапазону. Інтерфейс структуровано у вигляді набору панелей (панелей даних), кожна з яких відображає окрему метрику або групу пов'язаних метрик, а верхня частина екрана містить елементи керування часовим діапазоном та періодичністю автоматичного оновлення даних. Приклад інтерфейсу дашборду наведено на рисунку 2.3.

Основним елементом інтерфейсу SIEM-платформи Graylog, що використовується адміністратором безпеки для аналізу подій, є розділ Search, що надає текстове поле для формування пошукових запитів мовою запитів Graylog, фільтр за переліком потоків та інтерактивний графік розподілу кількості повідомлень у часі з можливістю групування за визначеним полем, зокрема за полем event_category, сформованим конвеєром обробки повідомлень. Розділ Streams надає інтерфейс керування правилами класифікації подій, розділ System → Pipelines – інтерфейс редагування правил конвеєра обробки повідомлень мовою правил Graylog, а розділ Dashboards – інтерфейс створення та перегляду інформаційних панелей, що об'єднують декілька віджетів пошуку та візуалізації на одному екрані, приклад якого – інформаційна панель SOC Security Overview – наведено на рисунку 2.8.

Узагальнений перелік основних розділів веб-інтерфейсу Graylog, що використовуються адміністратором безпеки у щоденній роботі з програмним комплексом, наведено у таблиці 3.1.

Таблиця 3.1

Перелік основних розділів веб-інтерфейсу Graylog

Розділ інтерфейсу	Призначення
Search	Формування довільних пошукових запитів, перегляд та фільтрація журналів подій за часовим діапазоном і потоками
Streams	Перегляд та редагування потоків класифікації подій (під-розділ 2.3)

		Шатілов М. В.			ККІКБ.КР.Б – 125 – 12 – ПЗ	Арк.
		Воротніков В. В.				
Змн.	Арк.	№ докум.	Підпис	Дата		57

Розділ інтерфейсу	Призначення
System → Pipelines	Редагування правил конвеєра Security Event Tagging (під-розділ 2.3)
System → Inputs	Керування вхідними модулями прийому журналів (Syslog UDP)
System → Users and Teams	Керування обліковими записами та токенами доступу до REST API
Dashboards	Перегляд інформаційної панелі SOC Security Overview
Alerts	Перегляд історії спрацювання умов сповіщень Graylog

Інтерфейс веб-консолі Prometheus, доступний за адресою <http://10.0.0.20:9090>, використовується переважно для діагностичних цілей – перевірки статусу підключення цільових вузлів моніторингу (розділ Status → Targets) та поточного стану правил генерації тривог (розділ Alerts), що дозволяє адміністратору оперативно діагностувати причину відсутності очікуваних метрик чи сповіщень без звернення безпосередньо до конфігураційних файлів. Аналогічну діагностичну функцію щодо стану маршрутизації сповіщень виконує веб-інтерфейс Alertmanager, доступний за адресою <http://10.0.0.20:9093>.

Інтерфейс взаємодії з програмним модулем автоматизованого реагування реалізований засобами месенджера Telegram та складається з текстових повідомлень із форматуванням HTML (виділення жирним шрифтом ключової інформації, моноширинний формат для відображення мережевих адрес) та інтерактивних кнопок, прикріплених до повідомлення засобами вбудованої клавіатури Telegram (inline keyboard). Кожне сповіщення про подію безпеки містить тип виявленої загрози, мережеву адресу джерела та короткий витяг з тексту первинного журналу події, а також, залежно від того, чи перевищено порогове значення автоматичного блокування, одну або дві кнопки дії.

Натискання кнопки в інтерфейсі Telegram призводить до миттєвої зміни тексту повідомлення засобом методу `edit_message_text` програмного інтерфейсу Telegram Bot API – кнопки видаляються, а текст повідомлення замінюється на під-

		Шатілов М. В.			ККІКБ.КР.Б – 125 – 12 – ПЗ	Арк.
		Воротніков В. В.				
Змн.	Арк.	№ докум.	Підпис	Дата		58

твердження виконаної дії (наприклад, « IP 192.168.50.20 розблоковано вручну»), що унеможливлює повторне натискання тієї ж кнопки та забезпечує адміністратору наочне підтвердження результату виконаної дії без необхідності звернення до додаткових журналів. Додатковим елементом інтерфейсу взаємодії є підтримка текстових команд /status та /unblock, що вводяться адміністратором безпосередньо у полі введення повідомлення месенджера та обробляються програмним модулем як звичайні текстові команди бота, попередньо зареєстровані у диспетчері бібліотеки python-telegram-bot.

3.3 Порядок роботи системи та тестовий приклад

Перевірка коректності функціонування розробленого програмного комплексу виконувалася шляхом послідовного моделювання характерних кіберінцидентів із віртуальної машини Kali Linux, розташованої у публічному сегменті мережі vlan-public, що відтворює дії зовнішнього зловмисника відносно цільового вузла інфраструктури. Загальний порядок тестового сценарію охоплює виконання атаки засобами спеціалізованого інструментарію, спостереження за послідовністю спрацювання компонентів виявлення (Snort, Graylog) та перевірку коректності реакції програмного модуля автоматизованого реагування.

Розглянемо детальний тестовий приклад на основі сценарію виявлення сканування мережевих портів. На першому етапі з командного рядка віртуальної машини Kali Linux виконується команда утиліти nmap у режимі сканування з прапором SYN, спрямована на мережеву адресу цільового вузла в сегменті vlan-public. Результати проведення цієї імітації розвідувальної діяльності зловмисника відображено на рисунку 3.1. Підсистема Snort, що аналізує трафік інтерфейсу enp0s9 цільового вузла, реєструє перевищення порогового значення, визначеного правилом ідентифікатора 1000004 (п'ять пакетів з прапором SYN від однієї адреси джерела за п'ять секунд), та формує запис у журналі повідомлень alert_syslog, що пересилається службою rsyslog на сервер SIEM/SOC. Приклад запису про зафіксовану подію в журналі наведено на рисунку 3.2.

		Шатілов М. В.			ККІКБ.КР.Б – 125 – 12 – ПЗ	Арк.
		Воротніков В. В.				
Змн.	Арк.	№ докум.	Підпис	Дата		59

```
(kali@kali)-[~]
$ nmap -sS -p 1-1000 192.168.50.10
Starting Nmap 7.98 ( https://nmap.org ) at 2026-06-22 05:19 -0400
Nmap scan report for 192.168.50.10
Host is up (0.00048s latency).
Not shown: 998 closed tcp ports (reset)
PORT      STATE SERVICE
22/tcp    open  ssh
80/tcp    open  http
MAC Address: 08:00:27:4E:F6:41 (Oracle VirtualBox virtual NIC)

Nmap done: 1 IP address (1 host up) scanned in 1.07 seconds
```

Рисунок 3.1 – Виконання сканування портів засобом nmap з віртуальної машини Kali Linux

```
serveradmin@target-server:~$ sudo tail -10 /var/log/snort/snort.alert.fast
06/22-09:19:04.069402 [жжж] [1:1000004:1] Possible NMAP Port Scan [жжж] [Priority: 0] {TCP} 192.168.50.20:58416 -> 192.168.50.10:837
06/22-09:19:04.071585 [жжж] [1:1000004:1] Possible NMAP Port Scan [жжж] [Priority: 0] {TCP} 192.168.50.20:58416 -> 192.168.50.10:57
06/22-09:19:04.073133 [жжж] [1:1000004:1] Possible NMAP Port Scan [жжж] [Priority: 0] {TCP} 192.168.50.20:58416 -> 192.168.50.10:988
06/22-09:19:04.074763 [жжж] [1:1000004:1] Possible NMAP Port Scan [жжж] [Priority: 0] {TCP} 192.168.50.20:58416 -> 192.168.50.10:247
06/22-09:19:04.076791 [жжж] [1:1000004:1] Possible NMAP Port Scan [жжж] [Priority: 0] {TCP} 192.168.50.20:58416 -> 192.168.50.10:659
06/22-09:19:04.078608 [жжж] [1:1000004:1] Possible NMAP Port Scan [жжж] [Priority: 0] {TCP} 192.168.50.20:58416 -> 192.168.50.10:377
06/22-09:19:04.080822 [жжж] [1:1000004:1] Possible NMAP Port Scan [жжж] [Priority: 0] {TCP} 192.168.50.20:58416 -> 192.168.50.10:348
06/22-09:19:04.082169 [жжж] [1:1000004:1] Possible NMAP Port Scan [жжж] [Priority: 0] {TCP} 192.168.50.20:58416 -> 192.168.50.10:867
06/22-09:19:04.084114 [жжж] [1:1000004:1] Possible NMAP Port Scan [жжж] [Priority: 0] {TCP} 192.168.50.20:58416 -> 192.168.50.10:969
06/22-09:19:04.086094 [жжж] [1:1000004:1] Possible NMAP Port Scan [жжж] [Priority: 0] {TCP} 192.168.50.20:58416 -> 192.168.50.10:902
serveradmin@target-server:~$
```

Рисунок 3.2 – Запис спрацювання правила Snort у журналі snort.alert.fast

На другому етапі вхідний модуль Graylog приймає надісланий запис журналу, після чого конвеєр обробки повідомлень Security Event Tagging, описаний у підрозділі 2.3, виконує перевірку відповідності тексту повідомлення правилу Tag Port Scan і додає до повідомлення поле event_category зі значенням Port Scan. Класифіковане повідомлення додатково потрапляє до потоку [SIEM] Snort IDS Alerts відповідно до правила фільтрації, наведеного у таблиці 2.5.

≡ All Messages

timestamp	source	event_category
2026-06-22 09:19:04.000	target-server	Port Scan
target-server snort[861]: [1:1000004:1] Possible NMAP Port Scan {TCP} 192.168.50.20:58416 -> 192.168.50.10:840		
2026-06-22 09:19:04.000	target-server	Port Scan
target-server snort[861]: [1:1000004:1] Possible NMAP Port Scan {TCP} 192.168.50.20:58416 -> 192.168.50.10:326		
2026-06-22 09:19:04.000	target-server	Port Scan
target-server snort[861]: [1:1000004:1] Possible NMAP Port Scan {TCP} 192.168.50.20:58416 -> 192.168.50.10:87		
2026-06-22 09:19:04.000	target-server	Port Scan
target-server snort[861]: [1:1000004:1] Possible NMAP Port Scan {TCP} 192.168.50.20:58416 -> 192.168.50.10:294		
2026-06-22 09:19:04.000	target-server	Port Scan
target-server snort[861]: [1:1000004:1] Possible NMAP Port Scan {TCP} 192.168.50.20:58416 -> 192.168.50.10:606		

Рисунок 3.3 – Класифіковане повідомлення про сканування портів у інтерфейсі Graylog Search

На третьому етапі програмний модуль автоматизованого реагування, що виконує періодичне опитування програмного інтерфейсу Graylog, отримує класифіковане повідомлення під час наступного циклу опитування (не пізніше 10 секунд після його появи в системі) та визначає мережеву адресу джерела атаки засобом аналізу тексту повідомлення за регулярним виразом. Оскільки порогове значення для категорії Port Scan встановлено рівним одній події (таблиця 2.8), модуль ухвалює рішення про автоматичне блокування виявленої адреси без очікування додаткових подій.

```
serveradmin@siem-server:~/soc-bot$ python3 soc_bot.py
/home/serveradmin/.local/lib/python3.12/site-packages/telegram/utils/request.py:
using upstream urllib3. This is allowed but not supported by python-telegram-b
warnings.warn(
2026-06-22 09:12:32,229 [INFO] Запуск опитування Graylog...
2026-06-22 09:12:32,233 [INFO] SOC Bot запущено. Очікування подій...
2026-06-22 09:12:32,437 [INFO] Scheduler started
2026-06-22 09:13:24,469 [INFO] IP 192.168.50.20 заблоковано на Target сервері.
```

Рисунок 3.4 – Журнал виконання програмного модуля під час обробки події Port Scan

На завершальному, четвертому етапі модуль виконує віддалене підключення до цільового вузла засобом протоколу SSH та додає правило фільтрації трафіку до таблиці INPUT засобами утиліти iptables, після чого надсилає адміністратору безпеки сповіщення в месенджер Telegram з інформацією про виконане блокування та інтерактивною кнопкою скасування дії. Загальний час від моменту початку сканування до отримання адміністратором сповіщення про виконане блокування під час експериментальної перевірки не перевищував 15 секунд, що включає час виявлення атаки засобами Snort, час пересилання журналу через rsyslog та Graylog, час очікування наступного циклу опитування модуля (до 10 секунд) та час виконання віддаленої команди блокування.

```
serveradmin@target-server:~$ sudo iptables -L INPUT -n
Chain INPUT (policy ACCEPT)
target      prot opt source                destination
DROP        0    -- 192.168.50.20          0.0.0.0/0
serveradmin@target-server:~$
```

Рисунок 3.5 – Підтвердження застосованого правила блокування у таблиці iptables цільового вузла

		Шатілов М. В.			ККІКБ.КР.Б – 125 – 12 – ПЗ	Арк.
		Воротніков В. В.				
Змн.	Арк.	№ докум.	Підпис	Дата		61

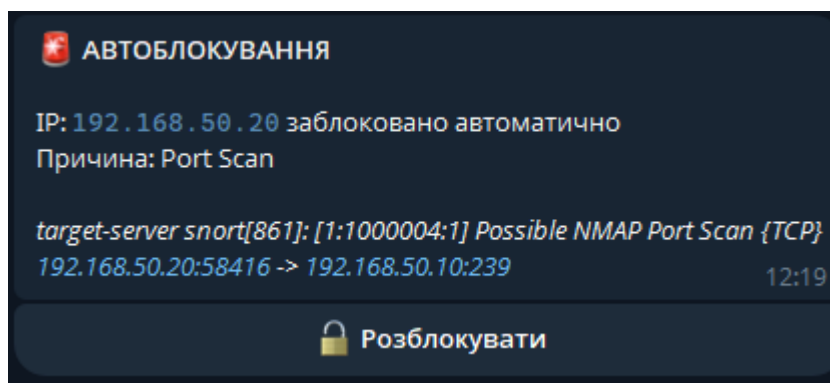


Рисунок 3.6 – Сповіщення адміністратора про виконане автоматичне блокування

Розглянемо також детальний тестовий приклад сценарію виявлення спроби впровадження SQL-коду, що ілюструє інший, відмінний від миттєвого блокування, режим роботи системи – накопичення подій у часовому вікні. З командного рядка Kali Linux формується серія HTTP-запитів із характерною лексичною конструкцією.

Команда формування серії запитів для тестування виявлення SQL-ін'єкції:

```
for i in 1 2 3; do
  curl -s "http://192.168.50.10/?id=1%27%20OR%20%271=1" \
    -o /dev/null
done
```

Кожен із трьох запитів містить URL-кодовану конструкцію `1' OR '1=1`, що відповідає умові правила Snort з ідентифікатором 1000002, розглянутого у підрозділі 2.4. Оскільки порогове значення для категорії SQL Injection встановлено рівним трьом подіям за 60 секунд (таблиця 2.8), перші дві події, отримані модулем автоматизованого реагування під час циклів опитування, призводять до накопичення лічильника у структурі ковзного часового вікна без виконання дії блокування, а лише третя подія, що досягає порогового значення, ініціює автоматичне блокування мережевої адреси джерела та надсилання відповідного сповіщення адміністратору. Зазначена особливість підтверджує коректність реалізації механізму накопичення подій, описаного у підрозділі 2.5, та практичну доцільність застосування порогового значення, що перевищує одиницю, для категорій подій з вищою ймовірністю поодинокого хибного спрацювання.

Окремої уваги потребує тестовий сценарій виявлення спроби підбору пароля протоколу SSH, оскільки відповідна подія належить до категорії мережевих атак,

		Шатілов М. В.			ККІКБ.КР.Б – 125 – 12 – ПЗ	Арк.
		Воротніков В. В.				
Змн.	Арк.	№ докум.	Підпис	Дата		62

що передбачають мережеву адресу джерела, але формується не підсистемою Snort, а безпосередньо системним журналом auth.log цільового вузла. Послідовність невдалих спроб підключення формується командою.

Команда формування серії невдалих спроб підключення для тестування виявлення SSH Brute-force:

```
for i in 1 2 3 4 5; do
  ssh -o StrictHostKeyChecking=no \
    -o ConnectTimeout=3 \
    fakeuser@10.0.0.10 exit 2>/dev/null
done
```

Кожна невдала спроба підключення фіксується службою sshd у системному журналі auth.log у форматі «Failed password for invalid user fakeuser from IP-адреса port номер-порту ssh2». Потік [SIEM] SSH Brute-force Attempts, правило фільтрації якого наведено у таблиці 2.5, класифікує кожен такий запис, а конвеєр обробки повідомлень додає поле event_category зі значенням SSH Brute-force. Оскільки порогове значення для цієї категорії становить п'ять подій за 60 секунд, що відповідає кількості спроб у наведеній тестовій команді, очікуваним результатом є автоматичне блокування мережевої адреси джерела одразу після обробки п'ятої події модулем автоматизованого реагування, що й було підтверджено під час експериментальної перевірки.

Аналогічним чином перевірено працездатність комплексу для решти визначених категорій подій: сканування каталогів веб-застосунку (засобом звернення до шляхів /admin, /backup, /wp-login) та використання автоматизованих засобів сканування вразливостей Nikto і sqlmap. Для кожного із зазначених сценаріїв підтверджено відповідність фактичної поведінки системи очікуваній згідно з визначеними у таблиці 2.8 пороговими значеннями: події, що не досягли порогу, призводили до надходження інтерактивного сповіщення з кнопками вибору дії, тоді як події, що відповідали або перевищували порогове значення, призводили до автоматичного блокування без участі адміністратора.

		Шатілов М. В.			ККІКБ.КР.Б – 125 – 12 – ПЗ	Арк.
		Воротніков В. В.				
Змн.	Арк.	№ докум.	Підпис	Дата		63

3.4 Перевірка коректності функціонування та результати тестування

Окрему увагу під час тестування приділено перевірці коректності роботи механізмів, що забезпечують стійкість програмного комплексу до нетипових ситуацій. Найбільш суттєвим із виявлених недоліків початкової реалізації виявився ефект рекурсивного реагування модуля на власні дії: під час виконання командою блокування iptables через протокол SSH з привілеями sudo на цільовому вузлі формувалася системний запис журналу авторизації, що сам по собі відповідав умові правила класифікації Privilege Escalation, оскільки містив підрядок sudo. Як наслідок, кожна виконана модулем команда блокування породжувала нову подію, яка після обробки могла призвести до повторного, цього разу безпідставного, блокування мережевої адреси сервера SIEM/SOC – джерела команди.

Початковий (некоректний) варіант правила класифікації, що спричиняв рекурсивне реагування:

```
rule "Tag Privilege Escalation"
when
    contains(to_string($message.message), "sudo:")
then
    set_field("event_category", "Privilege Escalation");
end
```

Виправлений варіант правила додає перевірку наявності специфічного ідентифікатора модуля автентифікації pam_unix(sudo:auth) у поєднанні з текстом повідомлення про помилку автентифікації (authentication failure), що відповідає виключно невдалим спробам автентифікації при виконанні команди sudo і не задовольняється записами про успішне виконання команд із застосуванням підвищених привілеїв, якими є записи, що формуються під час штатної роботи самого модуля. Після внесення цього виправлення повторне виникнення хибного спрацювання, спричиненого діями самого модуля, було усунено, що підтверджено повторним виконанням тестового сценарію автоматичного блокування протягом тривалого часу спостереження без появи безпідставних подій Privilege Escalation.

Перевірено коректність функціонування переліку довірених адрес (білого списку): спроби формування HTTP-запитів з мережевої адреси робочої станції адміністратора, що входить до переліку довірених адрес, коректно не призводили до

		Шатілов М. В.			ККІКБ.КР.Б – 125 – 12 – ПЗ	Арк.
		Воротніков В. В.				
Змн.	Арк.	№ докум.	Підпис	Дата		64

блокування, незважаючи на класифікацію відповідних подій підсистемою виявлення вторгнень. Перевірено коректність функціонування захисту від масового блокування: при штучному формуванні понад десяти подій різних категорій з різних мережевих адрес у межах одного часового вікна тривалістю 60 секунд модуль припиняв виконання подальших автоматичних блокувань та надсилав адміністратору відповідне попередження, що підтверджує коректність реалізованого механізму захисту, описаного у підрозділі 2.5.

Перевірено коректність очищення накопиченої історії подій під час ручного розблокування мережевої адреси: до внесення відповідного виправлення повторне надходження хоча б однієї нової події тієї ж категорії від щойно розблокованої адреси протягом залишку часового вікна спостереження призводило до миттєвого повторного автоматичного блокування навіть за відсутності продовження атаки, що було усунено додаванням явного очищення накопиченої історії спрацювань під час виконання процедури розблокування. Перевірено стабільність ідентифікатора вузла Graylog (Node ID) після перезапуску контейнеризованих сервісів: застосування іменованого тому Docker для каталогу даних Graylog, описане у підрозділі 2.2, забезпечило збереження раніше створеного вхідного модуля прийому журналів подій після повторного запуску сервера SIEM/SOC.

Узагальнені результати тестування програмного комплексу за основними категоріями подій наведено у таблиці 3.2.

Таблиця 3.2

Узагальнені результати тестування програмного комплексу

Категорія подій	Інструмент моделювання	Очікувана поведінка	Фактичний результат
SSH Brute-force	повторні спроби підключення з невірним паролем	блокування після 5 подій за 60 с	відповідає очікуваному
SQL Injection	сформований HTTP-запит з лексичною конструкцією	блокування після 3 подій за 60 с	відповідає очікуваному
Port Scan	птар (режим SYN-сканування)	блокування після 1 події	відповідає очікуваному
Directory Scan	звернення до /admin, /backup, /wp-login	блокування після 5 подій за 60 с	відповідає очікуваному

		Шатілов М. В.			ККІКБ.КР.Б – 125 – 12 – ПЗ	Арк.
		Воротніков В. В.				
Змн.	Арк.	№ докум.	Підпис	Дата		65

Категорія подій	Інструмент моделювання	Очікувана поведінка	Фактичний результат
Scanner Detected	nikto, sqlmap	блокування після 1 події	відповідає очікуваному
Privilege Escalation	невдала спроба sudo	лише сповіщення (локальна подія)	відповідає очікуваному
Direct Root Access	невдала спроба su root	лише сповіщення (локальна подія)	відповідає очікуваному

Результати проведеного тестування підтверджують повну відповідність фактичної поведінки розробленого програмного комплексу очікуваній згідно з визначеною архітектурою та обґрунтованими у другому розділі роботи пороговими значеннями. Виявлені під час тестування недоліки реалізації (хибна класифікація службових подій, рекурсивне реагування на власні дії модуля, відсутність очищення історії подій при ручному розблокуванні, втрата ідентифікатора вузла Graylog після перезапуску контейнерів) було усунено шляхом уточнення умов правил класифікації, доповнення алгоритму обробки подій перевіркою на відповідність діям самого модуля та застосування іменованих томів Docker для забезпечення сталості конфігурації між перезапусками контейнеризованих сервісів.

Висновки до третього розділу

У результаті виконання третього розділу роботи описано порядок встановлення та налаштування розробленого програмного комплексу, що охоплює підготовку віртуальних машин тестового стенда, розгортання контейнеризованих сервісів сервера SIEM/SOC, встановлення агентів моніторингу та підсистеми виявлення вторгнень на цільовому вузлі, а також налаштування безпечної віддаленої взаємодії між компонентами засобом протоколу SSH з автентифікацією за криптографічним ключем.

Описано структуру інтерфейсу взаємодії адміністратора безпеки з програмним комплексом, що реалізується через веб-інтерфейси складових компонентів Grafana та Graylog, а також через інтерфейс месенджера Telegram з інтерактивними

		Шатілов М. В.			ККІКБ.КР.Б – 125 – 12 – ПЗ	Арк.
		Воротніков В. В.				
Змн.	Арк.	№ докум.	Підпис	Дата		66

кнопками та текстовими командами керування. Наведено детальний тестовий приклад обробки інциденту сканування мережевих портів, що ілюструє повний цикл взаємодії підсистем комплексу від моменту виявлення атаки до виконання автоматичного блокування та сповіщення адміністратора.

Проведено експериментальну перевірку працездатності програмного комплексу для семи основних категорій подій безпеки, що підтвердила відповідність фактичної поведінки системи очікуваній згідно з обґрунтованою у другому розділі архітектурою. У процесі тестування виявлено та усунено чотири недоліки початкової реалізації, пов'язані з класифікацією подій, стійкістю до рекурсивного реагування та сталістю конфігурації між перезапусками контейнеризованих сервісів, що підвищило надійність функціонування розробленого програмного комплексу в цілому.

		Шатілов М. В.			ККІКБ.КР.Б – 125 – 12 – ПЗ	Арк.
		Воротніков В. В.				
Змн.	Арк.	№ докум.	Підпис	Дата		67

ВИСНОВКИ

У кваліфікаційній роботі вирішено актуальне науково-прикладне завдання щодо розробки та практичної реалізації системи виявлення інцидентів та комплексного моніторингу інформаційної інфраструктури підприємства на основі програмних рішень з відкритим вихідним кодом. Відповідно до поставленої мети було успішно досягнуто всіх завдань, визначених на початку дослідження.

На етапі аналізу сучасного стану технологій моніторингу та існуючих аналогів було доведено, що повноцінна система захисту вимагає трирівневого підходу (метрики, події, трафік) із додаванням механізмів автоматизованого реагування (SOAR). За результатами порівняльного аналізу було обґрунтовано вибір інструментальних засобів: для підприємств малого та середнього бізнесу найбільш раціональним є використання платформи Graylog як альтернативи комерційним SIEM-системам, зв'язки Prometheus та Grafana для моніторингу метрик, а також системи Snort для виявлення мережових вторгнень.

Відповідно до завдань практичної реалізації, було розроблено архітектуру тестового стенда із розмежуванням мережових сегментів, що гарантує повну ізоляцію мережі управління від публічного доступу та забезпечує безпечне моделювання атак. На базі цієї топології успішно розгорнуто та налаштовано підсистему збору метрик на основі Prometheus зі створенням шести правил генерації тривоги. Одночасно було налаштовано SIEM-платформу Graylog для централізованого збору подій безпеки, де завдяки впровадженню механізмів потоків (Streams) та дев'яти конвеєрних правил обробки повідомлень (Pipelines) досягнуто точної автоматичної класифікації інцидентів безпосередньо на етапі їх надходження.

Вирішуючи завдання контролю мережового трафіку, розгорнуто систему виявлення вторгнень Snort, для якої розроблено набір із десяти користувацьких сигнатурних правил, що детектують основні вектори атак на веб-застосунки (SQL-ін'єкції, сканування портів тощо). Для забезпечення активного захисту було розроблено програмний модуль автоматизованого реагування мовою Python. Він реалізував інтеграцію з Graylog REST API, оцінку загроз методом ковзного часового вікна,

		Шатілов М. В.			ККІКБ.КР.Б – 125 – 12 – ПЗ	Арк.
		Воротніков В. В.				68
Змн.	Арк.	№ докум.	Підпис	Дата		

інтерактивну взаємодію з адміністратором через месенджер Telegram та функцію віддаленого блокування шкідливих мережевих адрес засобами утиліти iptables.

Підсумковим етапом роботи стала комплексна експериментальна перевірка працездатності розробленого комплексу. Шляхом моделювання семи категорій кіберінцидентів з віртуальної машини Kali Linux було повністю підтверджено коректність функціонування всіх підсистем. Під час тестування також було успішно усунено виявлені інженерні проблеми, зокрема ефект рекурсивного реагування модуля на власні команди блокування, що дозволило досягти стабільної роботи системи.

Практичне значення одержаних результатів полягає в тому, що створений апаратно-програмний комплекс є повністю працездатним, гнучким у налаштуванні та готовим до впровадження в IT-інфраструктуру реальних підприємств. Він дозволяє суттєво підвищити загальний рівень захищеності мережі без витрат на ліцензування комерційного програмного забезпечення. Подальшим напрямком розвитку створеної системи може бути інтеграція алгоритмів машинного навчання для виявлення поведінкових аномалій (UEBA), а також масштабування архітектури з використанням платформ оркестрації контейнерів.

		Шатілов М. В.			ККІКБ.КР.Б – 125 – 12 – ПЗ	Арк.
		Воротніков В. В.				
Змн.	Арк.	№ докум.	Підпис	Дата		69

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Chuvakin A., Schmidt K., Phillips C. Logging and Log Management: The Authoritative Guide to Understanding the Concepts Surrounding Logging. Waltham : Syngress, 2012. 544 p.
2. Завгородній В. В. та ін. Сучасні архітектури систем управління подіями та інформаційною безпекою (SIEM). *Захист інформації*. 2021. Т. 23, № 2. С. 45–52.
3. Конохова О. В., та ін. Проектування центрів операцій з безпеки (SOC) для корпоративних мереж. *Безпека інформації*. 2022. Т. 28, № 1. С. 12–19.
4. ДСТУ ISO/IEC 27035-1:2019. Інформаційні технології. Метод захисту. Керування інцидентами інформаційної безпеки. Частина 1. Основні принципи. [Чинний від 2020-01-01]. Київ : ДП "УкрНДНЦ", 2019. 24 с.
5. Грайворонський М. В., Новіков О. М. Безпека інформаційно-комунікаційних систем. Київ : Видавнича група BVH, 2009. 608 с.
6. Корченко О. Г. Системи захисту інформації : навч. посібник. Київ : НАУ, 2004. 376 с.
7. Turnbull J. Monitoring with Prometheus. Brooklyn : Turnbull Press, 2018. 353 p.
8. Graylog: Pipelines – Rules and Functions. *Documentation*. URL: https://go2docs.graylog.org/current/making_sense_of_your_log_data/pipelines.html (дата звернення: 15.06.2026).
9. Splunk Enterprise Security: Security content and use cases. *Documentation*. URL: <https://docs.splunk.com/Documentation/ES/latest/Admin/UseCases> (дата звернення: 15.06.2026).
10. IBM QRadar: Supported DSMs and Event Collection. *Documentation*. URL: <https://www.ibm.com/docs/en/qsip/latest?topic=dsm-supported-dsms> (дата звернення: 15.06.2026).
11. Orebaugh A., Binde G., Babbin C. Snort IDS and IPS Toolkit. Burlington : Syngress, 2006. 800 p.
12. Snort 3 Rule Writing Guide: Payload Detection Rules. *Documentation*. URL: <https://docs.snort.org/rules/payload> (дата звернення: 15.06.2026).

		Шатілов М. В.			ККІКБ.КР.Б – 125 – 12 – ПЗ	Арк.
		Воротніков В. В.				70
Змн.	Арк.	№ докум.	Підпис	Дата		

13. Suricata User Guide: Deep Packet Inspection. *Documentation*. URL: <https://docs.suricata.io/en/latest/rules/payload-keywords.html> (дата звернення: 15.06.2026).
14. Wazuh: PCI DSS Compliance Monitoring. *Documentation*. URL: <https://documentation.wazuh.com/current/compliance/pci-dss/index.html> (дата звернення: 15.06.2026).
15. Suricata vs Snort: Performance and Architecture Comparison. Open Information Security Foundation Technical Report, 2023. 18 p.
16. Docker Compose: Networking in Compose. *Documentation*. URL: <https://docs.docker.com/compose/networking/> (дата звернення: 15.06.2026).
17. MongoDB 5.0 Release Notes – x86_64 Architecture Requirements (AVX). *Documentation*. URL: https://www.mongodb.com/docs/manual/release-notes/5.0/#x86_64-architecture-requirements (дата звернення: 15.06.2026).
18. Prometheus Node Exporter: Supported Collectors. *Documentation*. URL: <https://prometheus.io/docs/guides/node-exporter/#enabled-by-default> (дата звернення: 15.06.2026).
19. OpenSearch: Index mapping and management. *Documentation*. URL: <https://opensearch.org/docs/latest/opensearch/index-data/> (дата звернення: 15.06.2026).
20. python-telegram-bot: CallbackQueryHandler module. API Reference. URL: <https://docs.python-telegram-bot.org/en/v13.15/telegram.ext.callbackquery-handler.html> (дата звернення: 15.06.2026).
21. NIST Special Publication 800-61 Revision 2. Computer Security Incident Handling Guide. National Institute of Standards and Technology. Gaithersburg, MD, 2012. 79 p.
22. OpenSSH: sshd_config - Public Key Authentication. *Manual Pages*. URL: https://man.openbsd.org/sshd_config#PubkeyAuthentication (дата звернення: 15.06.2026).
23. NIST Special Publication 800-115. Technical Guide to Information Security Testing and Assessment. National Institute of Standards and Technology. Gaithersburg, MD, 2008. 80 p.

		Шатілов М. В.			ККІКБ.КР.Б – 125 – 12 – ПЗ	Арк.
		Воротніков В. В.				
Змн.	Арк.	№ докум.	Підпис	Дата		71

ДОДАТКИ

		Шатілов М. В.			ККІКБ.КР.Б – 125 – 12 – ПЗ	Арк.
		Воротніков В. В.				
Змн.	Арк.	№ докум.	Підпис	Дата		72

Лістинг конфігураційного файлу розгортання інфраструктури тестового стенда (docker-compose.yml)

docker-compose.yml:

```
version: '3'

services:
  mongodb:
    image: mongo:4.4
    restart: unless-stopped
    volumes:
      - mongodb_data:/data/db
    networks:
      - monitoring

  opensearch:
    image: opensearchproject/opensearch:1.3.7
    environment:
      - discovery.type=single-node
      - DISABLE_INSTALL_DEMO_CONFIG=true
      - DISABLE_SECURITY_PLUGIN=true
      - OPENSEARCH_JAVA_OPTS=-Xms1g -Xmx1g
    ulimits:
      memlock:
        soft: -1
        hard: -1
    ports:
      - "9200:9200"
    restart: unless-stopped
    volumes:
      - os_data:/usr/share/opensearch/data
    networks:
      - monitoring

  graylog:
    image: graylog/graylog:4.3
    environment:
      - GRAYLOG_PASSWORD_SECRET=somepasswordsecret12345678901234567890
      - GRAYLOG_ROOT_PASSWORD_SHA2=8c6976e5b5410415bde908bd4dee15dfb167a9c873fc4bb8a81f6f2ab448a918
      - GRAYLOG_HTTP_BIND_ADDRESS=0.0.0.0:9000
      - GRAYLOG_MONGODB_URI=mongodb://mongodb:27017/graylog
      - GRAYLOG_ELASTICSEARCH_HOSTS=http://opensearch:9200
    ports:
      - "9000:9000"
      - "1514:1514/udp"
      - "1514:1514/tcp"
    volumes:
      - graylog_data:/usr/share/graylog/data
    depends_on:
      - opensearch
      - mongodb
    restart: unless-stopped
```

		Шатілов М. В.			ККІКБ.КР.Б – 125 – 12 – ПЗ	Арк.
		Воротніков В. В.				
Змн.	Арк.	№ докум.	Підпис	Дата		73

```

networks:
  - monitoring

prometheus:
  image: prom/prometheus:latest
  volumes:
    - ./prometheus.yml:/etc/prometheus/prometheus.yml
    - ./alerts.yml:/etc/prometheus/alerts.yml
    - prometheus_data:/prometheus
  ports:
    - "9090:9090"
  restart: unless-stopped
  networks:
    - monitoring

grafana:
  image: grafana/grafana:latest
  environment:
    - GF_SECURITY_ADMIN_PASSWORD=admin123
  ports:
    - "3000:3000"
  volumes:
    - grafana_data:/var/lib/grafana
  restart: unless-stopped
  networks:
    - monitoring

alertmanager:
  image: prom/alertmanager:latest
  volumes:
    - ./alertmanager.yml:/etc/alertmanager/alertmanager.yml
    - ./templates.tmpl:/etc/alertmanager/templates.tmpl
  ports:
    - "9093:9093"
  restart: unless-stopped
  networks:
    - monitoring

volumes:
  mongodb_data:
  os_data:
  prometheus_data:
  grafana_data:
  graylog_data:

networks:
  monitoring:
    driver: bridge

```

		Шатілов М. В.			ККІКБ.КР.Б – 125 – 12 – ПЗ	Арк.
		Воротніков В. В.				
Змн.	Арк.	№ докум.	Підпис	Дата		74

Лістинг конфігураційних файлів підсистеми моніторингу та генерації тривоги (prometheus.yml, alerts.yml)

prometheus.yml:

```
global:
  scrape_interval: 15s
  evaluation_interval: 15s

rule_files:
  - "/etc/prometheus/alerts.yml"

alerting:
  alertmanagers:
    - static_configs:
        - targets:
            - alertmanager:9093

scrape_configs:
  - job_name: "prometheus"
    static_configs:
      - targets: ["localhost:9090"]

  - job_name: "target_server"
    static_configs:
      - targets: ["10.0.0.10:9100"]

  - job_name: "target_nginx"
    static_configs:
      - targets: ["10.0.0.10:9113"]
```

alerts.yml:

```
groups:
- name: security_and_system_alerts
  rules:
    - alert: InstanceDown
      expr: up == 0
      for: 1m
      labels:
        severity: critical
      annotations:
        summary: "Сервер недоступний"
        description: "Сервер {{ $labels.instance }} не відповідає."

    - alert: HighCPUUsage
      expr: (rate(node_cpu_seconds_total{mode="idle"}[1m])) * 100 > 85
      for: 3m
      labels:
        severity: warning
      annotations:
        summary: "Високе навантаження CPU"
        description: "CPU завантажено понад 85% протягом 3 хвилин."

    - alert: HighMemoryUsage
      expr: (node_memory_MemTotal_bytes - node_memory_MemAvailable_bytes) / node_memory_MemTotal_bytes * 100 > 90
      for: 3m
      labels:
```

		Шатілов М. В.			ККІКБ.КР.Б – 125 – 12 – ПЗ	Арк.
		Воротніков В. В.				
Змн.	Арк.	№ докум.	Підпис	Дата		75

```

        severity: warning
    annotations:
        summary: "Критичний рівень RAM"
        description: "Використовується понад 90% оперативної пам'яті."

- alert: DiskSpaceRunningOut
  expr:
    (node_filesystem_free_bytes{fstype!~"tmpfs|fuse.lxcfs|squashfs"}
    node_filesystem_size_bytes{fstype!~"tmpfs|fuse.lxcfs|squashfs"}) * 100 < 10
  for: 5m
  labels:
    severity: warning
  annotations:
    summary: "Закінчується місце на диску"
    description: "Залишилось менше 10% вільного місця."

- alert: HighNetworkTraffic
  expr: rate(node_network_receive_bytes_total[1m]) > 10000000
  for: 2m
  labels:
    severity: warning
  annotations:
    summary: "Аномальний мережевий трафік"
    description: "Можливий DDoS або витік даних."

- alert: UnauthorizedReboot
  expr: time() - node_boot_time_seconds < 300
  for: 1m
  labels:
    severity: critical
  annotations:
    summary: "Сервер перезавантажено"
    description: "Аптайм менше 5 хвилин."

- alert: HTTP404Spike
  expr: rate(nginx_http_requests_total{status="404"}[1m]) > 10
  for: 2m
  labels:
    severity: warning
  annotations:
    summary: "Спайк HTTP 404"
    description: "Можливе сканування директорій."

```

		Шатілов М. В.			ККІКБ.КР.Б – 125 – 12 – ПЗ	Арк.
		Воротніков В. В.				
Змн.	Арк.	№ докум.	Підпис	Дата		76

Програмний код модуля автоматизованого реагування на інциденти

```

soc_bot.py:
import time
import json
import logging
import subprocess
import threading
from collections import defaultdict, deque
from datetime import datetime, timedelta

import requests
from telegram import Update, InlineKeyboardButton, InlineKeyboardMarkup
from telegram.ext import Updater, CommandHandler, CallbackQueryHandler,
CallbackContext

import config

logging.basicConfig(
    format="%(asctime)s [%(levelname)s] %(message)s",
    level=logging.INFO,
)
log = logging.getLogger("soc_bot")

event_history = defaultdict(deque)

blocked_ips = {}

recent_blocks = deque()

last_seen_message_id = None

pending_actions = {}
pending_counter = 0

lock = threading.Lock()

def is_whitelisted(ip: str) -> bool:
    return ip in config.WHITELIST

def ssh_run(remote_command: str) -> tuple:

    cmd = [
        "ssh",
        "-i", config.SSH_KEY_PATH,
        "-o", "StrictHostKeyChecking=no",
        "-o", "ConnectTimeout=5",
        f"{config.TARGET_USER}@{config.TARGET_HOST}",

```

		Шатілов М. В.			ККІКБ.КР.Б – 125 – 12 – ПЗ	Арк.
		Воротніков В. В.				
Змн.	Арк.	№ докум.	Підпис	Дата		77

```

        remote_command,
    ]
    try:
        result = subprocess.run(cmd, capture_output=True, text=True,
                                timeout=15)
        if result.returncode == 0:
            return True, result.stdout.strip()
        else:
            return False, result.stderr.strip()
    except Exception as e:
        return False, str(e)

def block_ip(ip: str) -> bool:

    success, output = ssh_run(f"sudo iptables -C INPUT -s {ip} -j DROP
|| sudo iptables -I INPUT -s {ip} -j DROP")
    if success:
        log.info(f"IP {ip} заблоковано на Target сервері.")
        unblock_time = None
        if config.BLOCK_DURATION > 0:
            unblock_time = datetime.now() +
timedelta(seconds=config.BLOCK_DURATION)
        blocked_ips[ip] = unblock_time
        recent_blocks.append(datetime.now())
        return True
    else:
        log.error(f"Помилка блокування {ip}: {output}")
        return False

def unblock_ip(ip: str) -> bool:

    success, output = ssh_run(f"sudo iptables -D INPUT -s {ip} -j DROP")
    if success:
        log.info(f"IP {ip} розблоковано.")
        blocked_ips.pop(ip, None)
        keys_to_clear = [k for k in event_history.keys() if k[0] == ip]
        for k in keys_to_clear:
            event_history[k].clear()
        return True
    else:
        log.error(f"Помилка розблокування {ip}: {output}")
        return False

def too_many_recent_blocks() -> bool:

    now = datetime.now()
    while recent_blocks and (now - recent_blocks[0]).total_seconds() >
config.MAX_BLOCKS_WINDOW_SECONDS:
        recent_blocks.popleft()
    return len(recent_blocks) >= config.MAX_BLOCKS_PER_WINDOW

def check_threshold(ip: str, category: str) -> bool:

    threshold = config.THRESHOLDS.get(category)

```

		Шатілов М. В.			ККІКБ.КР.Б – 125 – 12 – ПЗ	Арк.
		Воротніков В. В.				
Змн.	Арк.	№ докум.	Підпис	Дата		78

```

        if not threshold:
            return False

        key = (ip, category)
        now = time.time()
        history = event_history[key]
        history.append(now)

        window = threshold["window"]
        while history and now - history[0] > window:
            history.popleft()

        return len(history) >= threshold["count"]

def send_telegram_message(text: str, reply_markup=None):
    url =
    f"https://api.telegram.org/bot{config.TELEGRAM_BOT_TOKEN}/sendMessage"
    payload = {
        "chat_id": config.TELEGRAM_CHAT_ID,
        "text": text,
        "parse_mode": "HTML",
    }
    if reply_markup:
        payload["reply_markup"] = reply_markup.to_dict()
    try:
        requests.post(url, json=payload, timeout=10)
    except Exception as e:
        log.error(f"Помилка надсилання в Telegram: {e}")

def notify_auto_block(ip: str, category: str, message: str):
    text = (
        f"🔒 <b>АВТОВЛОКУВАННЯ</b>\n\n"
        f"IP: <code>{ip}</code> заблоковано автоматично\n"
        f"Причина: {category}\n\n"
        f"<i>{message[:300]}</i>"
    )
    keyboard = InlineKeyboardMarkup([
        [InlineKeyboardButton("🔓", callback_data=f"unlock:{ip}")]]
    ])
    send_telegram_message(text, keyboard)

def notify_pending_decision(ip: str, category: str, message: str) ->
int:

    global pending_counter
    with lock:
        pending_counter += 1
        pid = pending_counter
        pending_actions[pid] = {"ip": ip, "category": category,
"message": message}

    text = (

```

		Шатілов М. В.			ККІКБ.КР.Б – 125 – 12 – ПЗ	Арк.
		Воротніков В. В.				79
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        f"⚠ <b>Можлива атака</b>\n\n"
        f"IP: <code>{ip}</code>\n"
        f"Тип: {category}\n\n"
        f"<i>{message[:300]}</i>"
    )
    keyboard = InlineKeyboardMarkup([
        [
            InlineKeyboardButton("🔒", callback_data=f"block:{pid}"),
            InlineKeyboardButton("✅", callback_data=f"ignore:{pid}"),
        ]
    ])
    send_telegram_message(text, keyboard)
    return pid

def button_callback(update: Update, context: CallbackContext):
    query = update.callback_query
    query.answer()
    data = query.data

    if data.startswith("unblock:"):
        ip = data.split(":", 1)[1]
        if unblock_ip(ip):
            query.edit_message_text(f"✅ IP {ip} розблоковано вру-
чну.")
        else:
            query.edit_message_text(f"❌ Помилка розблокування {ip}.")

    elif data.startswith("block:"):
        pid = int(data.split(":", 1)[1])
        action = pending_actions.pop(pid, None)
        if not action:
            query.edit_message_text(f"⚠ Цей запит вже неактуальний.")
            return
        if block_ip(action["ip"]):
            query.edit_message_text(f"🔒 IP {action['ip']} заблоковано
вручну (причина: {action['category']}).")
        else:
            query.edit_message_text(f"❌ Помилка блокування
{action['ip']}.")

    elif data.startswith("ignore:"):
        pid = int(data.split(":", 1)[1])
        action = pending_actions.pop(pid, None)
        if action:
            query.edit_message_text(f"✅ Подію з IP {action['ip']}
проігноровано.")
        else:
            query.edit_message_text(f"⚠ Цей запит вже неактуальний.")

def cmd_status(update: Update, context: CallbackContext):
    if not blocked_ips:

```

		Шатілов М. В.			ККІКБ.КР.Б – 125 – 12 – ПЗ	Арк.
		Воротніков В. В.				
Змн.	Арк.	№ докум.	Підпис	Дата		80

```

        update.message.reply_text("Наразі немає заблокованих IP.")
        return
    lines = ["🔒 <b>Заблоковані IP:</b>\n"]
    for ip, until in blocked_ips.items():
        if until:
            lines.append(f"• <code>{ip}</code> – до
{until.strftime('%Y-%m-%d %H:%M:%S')}")
        else:
            lines.append(f"• <code>{ip}</code> – назавжди")
    update.message.reply_text("\n".join(lines), parse_mode="HTML")

def cmd_unblock(update: Update, context: CallbackContext):

    if not context.args:
        update.message.reply_text("Використання: /unblock 1.2.3.4")
        return
    ip = context.args[0]
    if unblock_ip(ip):
        update.message.reply_text(f"✅ IP {ip} розблоковано.")
    else:
        update.message.reply_text(f"❌ Помилка розблокування {ip}.")

def auto_unblock_worker():

    while True:
        now = datetime.now()
        for ip in list(blocked_ips.keys()):
            until = blocked_ips.get(ip)
            if until and now >= until:
                if unblock_ip(ip):
                    send_telegram_message(f"🕒 IP <code>{ip}</code>
автоматично розблоковано (минув час блокування).")
                    time.sleep(30)

def fetch_new_graylog_messages():

    url = f"{config.GRAYLOG_URL}/api/search/universal/relative"
    params = {
        "query": "_exists_:event_category",
        "range": 60,
        "limit": 50,
        "sort": "timestamp:desc",
    }
    try:
        resp = requests.get(
            url,
            params=params,
            auth=(config.GRAYLOG_API_TOKEN, "token"),
            headers={"Accept": "application/json"},
            timeout=10,
        )
        resp.raise_for_status()
        data = resp.json()

```

		Шатілов М. В.			ККІКБ.КР.Б – 125 – 12 – ПЗ	Арк.
		Воротніков В. В.				
Змн.	Арк.	№ докум.	Підпис	Дата		81

```

        return data.get("messages", [])
    except Exception as e:
        log.error(f"Помилка запиту до Graylog: {e}")
        return []

def extract_ip_from_message(message_text: str) -> str:

    import re
    match = re.search(r"\b(?:\d{1,3}\.){3}\d{1,3}\b", message_text)
    return match.group(0) if match else None

def notify_local_event(category: str, message: str):
    text = (
        f"⚠ <b>Локальна подія безпеки</b>\n\n"
        f"Тип: {category}\n"
        f"<i>Блокування неможливе - подія відбувається безпосередньо  
на сервері.</i>\n\n"
        f"<i>{message[:300]}</i>"
    )
    send_telegram_message(text)

def process_graylog_messages():
    global last_seen_message_id
    messages = fetch_new_graylog_messages()

    for item in reversed(messages):
        msg = item.get("message", {})
        msg_id = msg.get("_id")
        category = msg.get("event_category")
        text = msg.get("message", "")
        if "iptables -I INPUT" in text or "iptables -D INPUT" in text:
            continue
        if not category or not msg_id:
            continue
        if msg_id == last_seen_message_id:
            continue

        if category in config.LOCAL_ONLY_CATEGORIES:
            notify_local_event(category, text)
            continue
        ip = extract_ip_from_message(text)
        if not ip:
            continue
        if is_whitelisted(ip):
            continue
        if ip in blocked_ips:
            continue

        exceeded = check_threshold(ip, category)

        if exceeded:
            if too_many_recent_blocks():
                send_telegram_message(
                    f"🔴 <b>Зафіксовано аномально багато спроб блоку-  
вання.</b>\n\n"
                    "Можлива масштабна атака (DDoS). Автоблокування  
тимчасово призупинено "
                )

```

		Шатілов М. В.			ККІКБ.КР.Б – 125 – 12 – ПЗ	Арк.
		Воротніков В. В.				
Змн.	Арк.	№ докум.	Підпис	Дата		82

```

        "для запобігання перевантаженню iptables. Переві-
рте систему вручну."
    )
    log.warning("Перевищено ліміт MAX_BLOCKS_PER_WINDOW -
блокування призупинено.")
    else:
        if block_ip(ip):
            notify_auto_block(ip, category, text)
        else:
            notify_pending_decision(ip, category, text)

    if messages:
        last_seen_message_id = messages[0].get("message",
        {}).get("_id", last_seen_message_id)

def graylog_polling_worker():
    log.info("Запуск опитування Graylog...")
    while True:
        try:
            process_graylog_messages()
        except Exception as e:
            log.error(f"Помилка в циклі опитування: {e}")
            time.sleep(config.POLL_INTERVAL)

def main():
    updater = Updater(token=config.TELEGRAM_BOT_TOKEN,
use_context=True)
    dp = updater.dispatcher

    dp.add_handler(CommandHandler("status", cmd_status))
    dp.add_handler(CommandHandler("unblock", cmd_unblock))
    dp.add_handler(CallbackQueryHandler(button_callback))

    threading.Thread(target=graylog_polling_worker,
daemon=True).start()
    threading.Thread(target=auto_unblock_worker, daemon=True).start()

    log.info("SOC Bot запущено. Очікування подій...")
    send_telegram_message("✅ SOC Bot запущено та готовий до монітори-
нгу.")

    updater.start_polling()
    updater.idle()

if __name__ == "__main__":
    main()

```

config.py:

```

TELEGRAM_BOT_TOKEN = "8823867690:AAHPhpU7U8j8vpIiKfKT3LB9DxBf1RgD290"
TELEGRAM_CHAT_ID = 788166169

GRAYLOG_URL = "http://localhost:9000"
GRAYLOG_API_TOKEN = "1i29rjecgpcv2n20qnnm6pn1dsg9dh40j2quap7cmvqh20atgu46"

```

		Шатілов М. В.			ККІКБ.КР.Б – 125 – 12 – ПЗ	Арк.
		Воротніков В. В.				
Змн.	Арк.	№ докум.	Підпис	Дата		83

```

TARGET_HOST = "10.0.0.10"
TARGET_USER = "serveradmin"
SSH_KEY_PATH = "/home/serveradmin/.ssh/soc_key"

WHITELIST = [
    "127.0.0.1",
    "10.0.0.10",
    "10.0.0.20",
    "10.0.0.30",
    "192.168.50.1",
]

THRESHOLDS = {
    "SQL Injection": {"count": 3, "window": 60},
    "Port Scan": {"count": 1, "window": 60},
    "Directory Scan": {"count": 5, "window": 60},
    "Scanner Detected": {"count": 1, "window": 60},
    "SSH Brute-force": {"count": 5, "window": 60},
    "Privilege Escalation": {"count": 3, "window": 60},
    "Direct Root Access": {"count": 3, "window": 60},
    "Unauthorized Account Creation": {"count": 1, "window": 60},
    "Cron Modification": {"count": 1, "window": 60},
}

LOCAL_ONLY_CATEGORIES = [
    "Privilege Escalation",
    "Direct Root Access",
    "Unauthorized Account Creation",
    "Cron Modification",
]

POLL_INTERVAL = 10

BLOCK_DURATION = 3600

MAX_BLOCKS_PER_WINDOW = 10
MAX_BLOCKS_WINDOW_SECONDS = 60

```

		Шатілов М. В.			ККІКБ.КР.Б – 125 – 12 – ПЗ	Арк.
		Воротніков В. В.				
Змн.	Арк.	№ докум.	Підпис	Дата		84

Лістинг користувацьких сигнатурних правил системи виявлення мережних вторгнень Snort

```

local.rules:
# $Id: local.rules,v 1.11 2004/07/23 20:15:44 bmc Exp $
# -----
# LOCAL RULES
# -----
# This file intentionally does not come with signatures.  Put your
local
# additions here.
# SQL Injection detection
alert tcp any any -> $HOME_NET 80 (msg:"SQL Injection Attempt - UNION
SELECT"; content:"UNION"; nocase; content:"SELECT">alert tcp any any ->
$HOME_NET 80 (msg:"SQL Injection Attempt - OR 1=1"; content:"OR"; nocase;
content:"1=1"; distance:>alert tcp any any -> $HOME_NET 80 (msg:"SQL
Injection Attempt - Quote"; content:"' OR '"; nocase; sid:1000003; rev:1;)

# Port Scan detection
alert tcp any any -> $HOME_NET any (msg:"Possible NMAP Port Scan";
flags:S; threshold: type threshold, track by_src, co>
# Directory/File scan
alert tcp any any -> $HOME_NET 80 (msg:"Directory Scan Attempt - admin";
content:"/admin"; nocase; sid:1000005; rev:1;)
alert tcp any any -> $HOME_NET 80 (msg:"Directory Scan Attempt - wp-
login"; content:"/wp-login"; nocase; sid:1000006; r>alert tcp any any ->
$HOME_NET 80 (msg:"Directory Scan Attempt - backup"; content:"/backup";
nocase; sid:1000007; rev:1>
# Known scanner User-Agents
alert tcp any any -> $HOME_NET 80 (msg:"Nikto Scanner Detected";
content:"Nikto"; http_header; sid:1000008; rev:1;)
alert tcp any any -> $HOME_NET 80 (msg:"SQLMap Scanner Detected";
content:"sqlmap"; http_header; sid:1000009; rev:1;)

# ICMP Flood
alert icmp any any -> $HOME_NET any (msg:"Possible ICMP Flood"; itype:8;
threshold: type threshold, track by_src, count>

```

		Шатілов М. В.			ККІКБ.КР.Б – 125 – 12 – ПЗ	Арк.
		Воротніков В. В.				
Змн.	Арк.	№ докум.	Підпис	Дата		85